



1401184329

**CRANFIELD INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF APPLIED COMPUTING AND MATHEMATICS**

**PhD THESIS**

**Academic Year 1990-1**

**S.G. PAVEY**

**A Building Modelling System for Environmental Design**

**Supervisor: M. Pratt**

**April 1991**

**ALL MISSING PAGES ARE BLANK**

**IN**

**ORIGINAL**

## ABSTRACT

Computerised systems have been developed to simulate the dynamic thermal performance of a building over a period of time, against a given weather pattern, with the aim of predicting the environmental and energy performance of the building. The usage of these systems has been limited, however due to the difficulty in describing the problem to be simulated to the system, and in interacting with the system. In particular the geometrical form of a building can be very tedious to input.

This thesis considers the graphical creation of a building model for environmental analysis, such that the necessary geometrical data can be automatically derived and input into the analysis. Current geometrical modelling techniques and the use of CAD in building design are considered. The geometrical and topological modelling requirements for the building model are established, and a data structure is derived and checked for sufficiency.

A possible modelling method is described for simple  $2\frac{1}{2}$  dimensional buildings, and uses the property that the floor plan of a building can be represented by a planar graph. The necessary extensions to the modelling method are considered to allow more complex 3 dimensional forms to be created. The assignment of attributes to the model, necessary for the thermal analysis, is also included.

Details are given of the functionality of a user interface, using an implementation of this modelling method. This includes a description of the interaction to create the model, and the graphical displays that are generated. This modelling method has also been used in the development of an interface to an architectural CAD system, such that the necessary building data can be transferred for analysis.

## CONTENTS

	Page
1. INTRODUCTION.....	1
2. CAD IN ENVIRONMENTAL DESIGN.....	3
2.1 Thermal Analysis of a Building.....	3
2.2 Simulation Processes.....	3
2.3 Use of Dynamic Simulation Programs.....	4
2.4 Use of Thermal Analysis in Building Design.....	6
2.5 A Computerised Design Tool for Thermal Analysis.....	7
2.6 Input of Building Geometry.....	8
2.7 Requirements for interactive graphical interface.....	9
2.8 Other Building Design Processes.....	9
2.9 Building Modeller.....	10
3. INPUT DATA TO THERMAL ANALYSIS.....	13
3.1 Shape and form of the building.....	13
3.1.1 Orientation.....	13
3.1.2 Slope.....	14
3.1.3 Area.....	14
3.1.4 Shading Type.....	14
3.1.5 Building Feature Type and Linking Zone.....	17
3.1.6 Zone Length, Height and Width.....	17
3.2 Thermal attributes of the constructions.....	18
3.2.1 Building Elements.....	18
3.2.2 Construction Database.....	18
3.2.3 Materials Database.....	18
3.3 Internal environment of the building.....	22
3.3.1 Internal Heat Gains.....	22
3.3.2 Environmental Control.....	23
3.3.3 Air Infiltration and Ventilation.....	23
3.4 External environment of the building.....	23
3.5 Plant specifications.....	24
3.6 Summary of Geometric Model.....	24
3.6.1 Zones and Spaces.....	24
3.6.2 Surfaces and Constructions.....	25
3.6.3 Zonal Arrangement.....	25
3.6.4 Requirements of Geometric Model.....	26
4. CURRENT CAD SYSTEMS AND MODELLING TECHNIQUES.....	29
4.1 2D Draughting System.....	29
4.2 2D Modelling.....	29
4.3 3D Modelling.....	30
4.4 Solid Modelling.....	31
4.5 Building Modelling.....	31
4.6 Use of Architectural CAD systems during the design process.....	33
4.7 Applicability of CAD systems to environmental design.....	34



5.	BOUNDARY REPRESENTATIONS IN SOLID MODELLING.....	37
5.1	Topology of Boundary Representation.....	37
5.1.1	2D Manifold.....	37
5.1.2	Connected Manifold.....	37
5.1.3	Bounded Surface.....	38
5.1.4	Closed and Orientable Surfaces.....	38
5.2	Euler-Poincare Formula.....	38
5.3	Embedded Graphs.....	39
5.4	Graph-based Data Structures.....	39
5.5	Definition of data records.....	40
5.5.1	Face loops.....	42
5.5.2	Hierarchical Structure of Object.....	42
5.5.3	Special case of single vertex loop.....	42
5.5.4	Wire edges and closed edges.....	42
5.5.5	Additional Backward and Sideway pointers.....	42
5.5.6	Description of Data Records.....	43
5.6	Access of topological relationships.....	45
5.7	Definition of Geometry.....	46
5.8	Euler Operators.....	46
5.9	Implementation of Euler Operators.....	47
5.9.1	Geometry.....	47
5.9.2	Input Parameters.....	47
5.9.3	Returned Parameters.....	50
5.9.4	Operation performed by a function.....	50
5.10	Components of modelling system.....	50
6.	BUILDING MODEL REPRESENTATION.....	53
6.1	Spaces and Constructions.....	53
6.2	Comparison of Model with Boundary Representation.....	53
6.3	Representation of Space by Embedded Graph.....	54
6.4	Topological constraints.....	56
6.5	Data Structures and Adjacency Relationships.....	59
6.6	Geometry of Building Model.....	59
6.7	Derivation of Data Structure.....	60
6.7.1	Construction and Space Relationships.....	61
6.7.2	Node and Vertex relationships.....	61
6.7.3	Node - Space Relationship.....	64
6.7.4	Node - Construction Relationships.....	67
6.7.5	Multi-edge - Edge Relationships.....	67
6.7.6	Space - Multi-edge Relationships.....	70
6.7.7	Multi-edge - Construction Relationships.....	71
6.7.8	Multi-edge - Node Relationships.....	71
6.7.9	Self relationships.....	74
6.8	Summary of Data Entities.....	75
6.8.1	Representation of Geometry.....	75
6.8.2	Definition of Data Records.....	76
7.	CREATION OF BUILDING DATA STRUCTURE.....	81
7.1	Building Geometry.....	82
7.2	Planar Graphs.....	82
7.3	Modelling Method.....	85
7.4	Creation of Planar Graph representing Floor.....	85
7.5	Creation of Building Model from Planar Graph.....	86
7.6	Result of Modelling Operation.....	87

8.	CREATION OF PLANAR GRAPH REPRESENTING FLOOR.....	89
8.1	Specification of Process.....	89
8.1.1	Checking for Coincident Edge.....	93
8.2	Attachment of Edges.....	97
8.3	Faces and Loops.....	97
8.3.1	Creation of a Face.....	99
8.3.2	Deletion of a Hole Loop.....	101
8.3.3	Geometrical Calculations for Loops and Faces.....	101
8.4	Tolerance of Graph.....	105
8.4.1	Finding Position of Point in Graph.....	108
8.4.2	Minimum edge length.....	108
8.4.3	Checking for Coincident Edges.....	111
8.4.4	Value of Graph Tolerance.....	111
9.	CREATION OF BUILDING MODEL FROM PLANAR GRAPH.....	113
9.1	Creation of Geometry and Nodes.....	113
9.2	Creation of Spaces and Constructions.....	114
9.3	Temporary Data Pointers.....	122
9.3.1	Graph Edges and Constructions.....	122
9.3.2	Graph vertices and Nodes.....	122
9.4	Use of winged-edge data structures.....	123
9.5	Specification of Functions performing Euler Operations.....	126
10.	FURTHER MODELLING REQUIREMENTS.....	129
10.1	Multi-floor Buildings.....	129
10.2	Spaces with complex 3D geometry.....	136
10.2.1	Spaces with varying heights.....	136
10.2.2	Courtyards.....	139
10.2.3	Sloping Roof Face.....	139
10.2.4	Spaces with more than one sloping face.....	139
10.2.5	'Merged' Spaces.....	142
10.2.6	Spaces extending over more than one floor.....	142
10.3	Creation of Complex 3D Geometry.....	145
10.3.1	Sweep process using a vertex sweep height and direction.....	145
10.3.2	Creation of Internal and External Wall Constructions.....	150
10.3.3	Deletion of Wall Construction to merge spaces.....	152
10.3.4	Creation of Extended Spaces.....	152
10.3.5	Representation of Space as set of Units.....	156
10.3.6	Intersection of Ceiling Face with Floor level.....	158
10.3.7	Modified Sweep Operation.....	161
10.3.8	Creation of Planar Graph from Building Model.....	162
10.4	Summary of Building Model Creation.....	163
11.	ASSIGNMENT OF MODEL ATTRIBUTES.....	165
11.1	Allocation of Construction Type.....	165
11.1.1	Construction Category Data Entity.....	166
11.1.2	Assignment of Category to constructions.....	166
11.1.3	Construction Thickness.....	168
11.2	Windows.....	180
11.2.1	Window Types.....	180
11.2.2	Representation of Window Instances.....	180
11.2.3	Creation of Window Construction.....	181
11.2.3.1	Derivation of Wall Window.....	182



11.2.3.2	Derivation of Roof Window.....	182
11.3	Zones.....	183
12.	INTERFACE TO CREATE FLOOR GRAPH.....	185
12.1	Creation of 2D Geometry.....	185
12.1.1	User Input of 2D points.....	185
12.1.2	Applying constraints to Input Points.....	187
12.1.3	Window Positioning.....	188
12.2	Graph Edge and Graph Face Data Entities.....	192
12.3	Assignment of Attributes.....	193
12.3.1	Height.....	194
12.3.2	Inclination.....	194
12.3.3	Offset Type.....	197
12.3.4	External Flag.....	197
12.3.5	Zones.....	197
12.3.6	Construction Types.....	198
12.4	Manipulation of Attributes with Euler Operations.....	198
12.5	Floor Editing Process.....	201
12.5.1	Defining a Digitiser Mapping.....	201
12.5.2	Copying between floors.....	204
12.5.3	3D Displays of Building Model.....	204
12.5.3.1	Wireline.....	205
12.5.3.2	Hidden Surface.....	205
12.5.3.3	Floor Plan.....	206
12.6	Interface to Architectural CAD.....	206
12.6.1	Transfer File.....	208
12.6.2	Access of Data from CAD Database.....	208
12.6.3	Data Interpretation.....	209
13.	INTERFACE TO ANALYSIS.....	219
13.1	Building Position.....	219
13.2	TAS Analysis Data.....	219
13.2.1	Zone Building Element Names.....	220
13.2.2	Zone Data.....	220
13.3	Generation of Analysis Data.....	221
13.3.1	Zone Volume and Floor Area.....	221
13.3.2	Zone Surface Data.....	221
13.4	Shading Evaluation.....	226
13.4.1	Creation of Site.....	226
13.4.2	Shading Calculations.....	227
13.4.3	Display of Shadows.....	232
13.4.4	Thermal Analysis using Shading Data.....	232
14.	CONCLUSION.....	235
	REFERENCES .....	239
	ACKNOWLEDGEMENTS .....	241
	APPENDIX A.....	243
	APPENDIX B.....	247
	APPENDIX C.....	257

## FIGURES

No.	Title	Page
2-1	Simulation Processes.....	5
2-2	Structure of Building Modelling System.....	11
3-1	Building Orientation & Zone Relative Orientation.....	15
3-2	Feature Shading.....	16
3-3	Building with two zones.....	19
3-4	Example of Zone surface data.....	20
3-5	Example of Building Elements.....	20
3-6	Example of a Construction.....	21
5-1	Relationships represented by a winged-edge.....	41
5-2	Euler operations.....	48
5-2	Euler operations (cont).....	49
6-1	Points on space boundaries.....	55
6-2	Representation of adjacent spaces by different embedded graphs.....	55
6-3	Construction represented by two co-incident faces.....	55
6-4	Faces of a Construction.....	57
6-5	Vertices of Node.....	57
6-6	Edges of a Multi-edge.....	58
6-7	Space - Constructions relationship.....	62
6-8	Construction - Spaces relationship.....	62
6-9	Reference pointers deriving Construction - Space Relationships.....	63
6-10	Node - Vertices Relationship.....	65
6-11	Space - Nodes relationship.....	66
6-12	Node - Spaces relationship.....	66
6-13	Construction - Nodes relationship.....	68
6-14	Node - Constructions relationship.....	68
6-15	Generation of 2 lists of vertices for the definition of a Multi-edge.....	69
6-16	Multi-edge - Spaces relationship.....	72
6-17	Construction - Multi-edges relationship.....	72
6-18	Multi-edge - Constructions relationship.....	73
7-1	Representation of floor plan by a planar graph.....	84
8-1	Wall divided into graph edges.....	90
8-2	Creation of vertex for first endpoint of wall.....	91
8-2	Creation of vertex for first endpoint of wall (cont).....	92
8-3	No intersection found: Make edge and vertex.....	94
8-4	Intersection with existing edges: Split edge.....	95
8-5	Vertices of same loop: Make an edge and face.....	95
8-6	Loops of vertices differ: Make edge, kill a loop.....	96
8-7	Examples of coincident edges.....	96
8-8	Finding the clockwise edge.....	98
8-9	Peripheral and hole loops.....	98
8-10	Moving hole loops between faces.....	100
8-11	Swapping hole and peripheral loops.....	100
8-12	Swapping hole and peripheral loops of external face.....	102



8-13	Joining peripheral and hole loops.....	102
8-14	Joining two hole loops.....	103
8-15	Calculation for peripheral and hole loops.....	104
8-16	Checking for point inside or outside of loop.....	106
8-17	Checking a point is in a face.....	107
8-18	Graph tolerance.....	109
8-19	Adjusting point 'on' edge to be 'coincident' with vertex.....	109
8-20	Points on more than one edge.....	110
8-21	Checking for coincidence of edge with line.....	112
9-1	Representation of nodes by a graph vertex.....	115
9-2	Representation of lower node.....	115
9-3	Creation of 2 face lamina to represent new space.....	116
9-4	Creation of Space Entity.....	118
9-5	Creation of references between Space Vertex and Node.....	119
9-6	Creation of side face.....	119
9-7	Creation of references between top vertices and node instances, and side faces and space.....	120
9-8	Construction entity.....	121
9-9	Temporary storage of pointers to node instances in graph edge entity.....	124
9-10	Determining required face/vertex instance.....	125
10-1	Floor/Ceiling constructions formed by joining floor models.....	130
10-2	Constructions created from floor faces of upper floor and ceiling faces of lower floor.....	131
10-3	Adjustment of flags on creation of new face.....	133
10-4	Adjustment of flags when moving loops.....	134
10-5	Adjustment of flags when edges are coincident.....	135
10-6	Creation of constructions from remaining external floor and ceiling faces.....	137
10-7	Division of face of higher space to create an internal and external wall construction.....	138
10-8	Creation of inclined ceiling face, by using different sweep heights for each vertex.....	140
10-9	Division of face by non-horizontal edge.....	140
10-10	Division of both faces by edges.....	141
10-11	Space with more than one sloping face.....	141
10-12	Creation of complex 3 dimensional shape by merging spaces.....	143
10-13	Space extending over more than one floor.....	144
10-14	Graph Representation of floor 2.....	144
10-15	Division of ceiling face at intersection with floor level.....	146
10-16	Result of sweep using height as assigned to vertex, and inclinations of adjoining graph edges.....	146
10-17	Calculation of normal of side face from direction of graph edge and assigned inclination.....	148
10-18	Calculation of swept point.....	148
10-19	Calculation of swept point P' for a colinear vertex at point P.....	149
10-20	Intersection of sweep vector of colinear vertex with 'up' edge of corner vertex.....	151
10-21	Check for well-formed side faces.....	151

10-22	Process for 'Delete Construction and Space' operation.....	153
10-23	'Split Edge' operation, implying modification to original wall construction.....	155
10-24	Representation of space extending over two floors by two space units.....	155
10-25	Representation of space as set of units.....	157
10-26	Example of space with sloping ceiling face intersecting a floor level.....	159
10-27	Division of space into space units.....	160
10-28	Process of Building Model Creation.....	164
11-1	Cross-section of space showing offset points at intersections of offset faces.....	170
11-2	Floor offset point at intersection of 2 offset wall planes and horizontal floor plane.....	171
11-3	Floor/ceiling offset points at intersection of offset wall and ceiling planes and horizontal floor plane.....	172
11-4	Ceiling offset point at intersection of two offset wall planes and offset ceiling plane.....	173
11-5	Adjustment of ceiling offset point below floor level.....	174
11-6	Adjustment of ceiling offset point below floor plane.....	175
11-7	Calculation of ceiling offset point when there is no ceiling face.....	177
11-8	Floor offset point of co-linear vertex.....	178
11-9	Placement of window in wall.....	184
11-10	Placement of window in roof.....	184
12-1	Transformation mapping from digitiser to world space.....	186
12-2	Adjustment of endpoint 'on an edge' when direction is set.....	189
12-3	Adjustment of line to parallel position when endpoint is 'at a vertex' and direction is set.....	190
12-4	Adjustment of line to parallel position when endpoint is 'at a vertex, startpoint is 'on an edge' and direction is set.....	190
12-5	Mismatch of endpoints when completing a loop with orthogonal lines.....	191
12-6	Extension of existing edge and adjustment of line to parallel position when both points are 'at a vertex' and direction is set.....	191
12-7	Height assignment by space.....	195
12-8	Height assignment by edge.....	195
12-9	Height assignment by vertex.....	196
12-10	Calculation of height of vertex from defined plane.....	196
12-11	Assignment of height attributes with 'add edge and vertex' operation.....	202
12-12	Assignment of height attributes with 'add edge and face ' or add edge and kill hole' operations.....	203
12-13	Assignment of height attributes with 'split edge' operation.....	203
12-14	Transfer process of building data from architectural CAD systems.....	207
12-15	Example of two connected walls with non-coincident end points.....	211
12-16	Using intersection point to determine the position of the wall endpoints.....	212



12-17	Using intersection point to determine a vertex is 'on the wall line' .....	213
12-18	Example of constraining vertex with two edges.....	213
12-19	Redefinition of wall line to attach to constraining vertex.....	215
12-20	Redefinition of wall line to attach to two constraining vertices.....	216
12-21	Example of two constraining vertices, with differing nearest wall lines.....	217
13-1	Generation of zone surface data from space face.....	222
13-2	Defining orientation of building element from offset type.....	225
13-3	Calculation of sun direction in building model co-ordinate system.....	228
13-4	Projection of face onto back face producing shadow .....	230
13-5	2D graph representing projection of face onto back face.....	230
13-6	Definition of area of window in sun.....	231

## TABLES

No.	Title	Page
3-1	Entry in Materials Database.....	22
3-2	Hourly Weather Data.....	23
5-1	Adjacency Relationships between Topological Entities.....	40
5-2	Edge Data Record.....	43
5-3	Object Data Record.....	43
5-4	Shell Data Record.....	43
5-5	Face Data Record.....	44
5-6	Loop Data Record.....	44
5-7	Vertex Data Record.....	44
6-1	Adjacency Relationships.....	59
6-2	Association of Geometry with Topological Entities.....	60
6-3	Building Data Record.....	76
6-4	Space Data Record.....	76
6-5	Construction Data Record.....	77
6-6	Node Instance Data Record.....	77
6-7	Point Data Record.....	77
6-8	Edge Data Record.....	78
6-9	Face Data Record.....	78
6-10	Loop Data Record.....	79
6-11	Vertex Data Record.....	79
9-1	Usage of winged-edge data structure.....	123
9-2	Euler Operations required for usages of winged-edge data.....	127
10-1	Data Record of Space represented as set of units.....	158
10-2	Spacelink Data Record.....	158
11-1	Default Building Element Allocation.....	165
11-2	Category Data Record.....	166
11-3	Space Data Record referencing Categories.....	167
11-4	Construction Data Record referencing Categories.....	168
11-5	Node Instance Data Record referencing offset point.....	179
11-6	Construction Data Record referencing offset type.....	179
11-7	Window Data Record.....	181
11-8	Construction Data Record referencing Windows.....	181
11-9	Space Data Record referencing zone.....	183
12-1	Graph Face Data Record.....	192
12-2	Graph Edge Data Record.....	193
12-3	Height Attribute Assignment to Graph Edge.....	199
12-4	Inclination Attribute Assignment to Graph Edge.....	199
12-5	Offset Type Assignment to Graph Edge.....	199
12-6	Wall Category Assignment to Graph Edge.....	200
12-7	Wall Window Assignment to Graph Edge.....	200
12-8	Zone Number Assignment to Graph Face.....	200
12-9	External Flag Assignment to Graph Face.....	200
12-10	Floor/Ceiling Categories Assignment to Graph Face.....	201
12-11	Floor/Ceiling Windows Assignment to Graph Face.....	201
12-12	Data Items defining Transferred Wall.....	209

## 1. INTRODUCTION

The energy and environmental performance of a building is predicted during the design process of that building. The analysis can influence such factors as the construction materials used, the heating and/or cooling capacity required for any plant, and ensures that any required energy targets are met.

Manual methods have been available for decades. The computerisation of these methods, and the development of more complex analyses, which take advantage of computers, have occurred within the last twenty years. Computerised energy analysis is now used by even small engineering companies.

Many of the computerised analysis methods in use are based on 'steady state' methods, which use the concept that the heat which is entering a building or is supplied to it, is balanced by the heat lost. The storage of the heat in the fabric of the building is ignored.

Over the last decade there has been more emphasis on research into energy use with the aim of conserving energy through improved design of buildings. Analysis systems have been developed which simulate the dynamic thermal performance of buildings, so demonstrating the way a building responds to fluctuating external weather conditions and the supply of heat or cooling to the building over a period of time.

However the application of these analysis systems has been limited, mainly due to the way the analysis software is made available to a user, usually an engineer. The emphasis on the development of these systems has been on the methods of analysis and their implementation, with less attention being paid to how the problem to be analysed is actually described and input into the analysis software. Their user interface has not been designed for ease of use, taking into account how an engineer works in practice, but as a means to get the required data to the analysis module. It has proved difficult, therefore, to use such a system as a design tool in the process of designing a building.

There have been different analytical approaches, resulting in the development of many computer programs. One such system is TAS - Thermal Analysis System, upon which the work in this thesis is based. Although originally developed as a research tool it is now commercially available.

This thesis is concerned with the development of a design tool, to allow greater ease of use of thermal analysis systems. The main emphasis has been to allow the building form to be described and input in a more natural and interactive way. This requires a building model to be created, from which the necessary geometric data for an analysis of the building is then derived. The geometric and topological modelling requirements and the implementation to create such a model are discussed in detail.



Chapter 2 introduces thermal analysis and dynamic simulation, and how it is currently applied. The limitations in application and the attributes of a possible computerised design tool are discussed, emphasising the importance of the input and representation of the building shape and form.

The data necessary to define a building model for analysis, as input into TAS, are described in detail in Chapter 3, from which the requirements of a geometric model of a building are derived, such that an analysis of the model is possible. However an investigation of current CAD as applied to building design and geometric modelling techniques, as discussed in Chapter 4, highlights a number of deficiencies in satisfying these requirements.

In Chapter 3, it is established that an unambiguous model of a building is required, therefore, solid modelling techniques are investigated, in Chapter 5, as a method of geometrical modelling with no ambiguity in the resulting model. The domain of objects modelled by a boundary representation and the data structures used are discussed, together with the methods used to create and access such models. The necessary components of a modelling system are therefore established.

Referring to the boundary representation, and given the building modelling requirements defined in Chapter 3, Chapter 6 discusses the representation of a building model in detail. Topological constraints are defined, and the sufficiency of a data representation is investigated, to enable the entities of a data structure to be derived.

Chapter 7 investigates a possible modelling method to create the desired building representation. The method is derived by considering that the form of a building is inherently  $2\frac{1}{2}$  dimensional, and that a 2 dimensional floor plan can be represented by a planar graph. This basic modelling method is divided into two stages, and these are described in detail in Chapters 8 and 9.

To allow more complex geometrical forms to be modelled establishes further modelling requirements, that is it is necessary that multi-floor buildings and 3 dimensional roof shapes can be modelled. Chapter 10 therefore defines these, and also considers the user interaction required when creating a model.

Attributes must also be included in the building representation, assigned to the data entities, to enable a thermal analysis of the building model. This extension of the data structure is considered in Chapter 11, together with the representation of windows in the building model.

The user interface requirements of the modelling system to produce the desired design tool are addressed in Chapter 12, and also a possible method to interface to an architectural CAD system is described.

Finally the interface to thermal analysis is discussed in Chapter 13, where the building model is accessed to generate the necessary input data for the thermal analysis. Shading calculations are also considered for the purpose of both thermal analysis and solar analysis.

## 2. CAD IN ENVIRONMENTAL DESIGN

### 2.1 Thermal Analysis of a Building

Thermal analysis systems that perform a dynamic simulation of a building, model the flow of heat through the mass of the building, together with a proposed plant control strategy, to predict the resulting temperatures and energy consumptions. More complex questions can be answered than with the steady state models:

- a) What and when are the peak heating or cooling loads for the building to specify the necessary capacity of the plant.
- b) What is the best control strategy for the plant to save energy, considering buildings which are heated intermittently, and what is the optimum start-up time.
- c) What construction materials should be used, what level of insulation should be employed and where, considering the intermittent heating of buildings and that not all rooms need to be heated to the same temperature.
- d) What geometric features should the building have, for shading purposes or in passive solar design, where storage of heat in the building fabric is desired, but not such that it causes overheating.

### 2.2 Simulation Processes

A building is modelled as a network of thermal resistances and capacitances. A computerised simulation needs to analyse the following processes against time:

- a) The transient conduction of heat through the building fabric.
- b) The shortwave solar radiation impinging on the external surfaces of the building, and through glazed surfaces to internal surfaces.
- c) The longwave radiation exchange between external surfaces and the sky.
- d) The longwave radiation exchange between internal surfaces.
- e) The convective process.
- f) Infiltration and ventilation.
- g) The control and response of the heating or cooling plant, dependent on operational times, location of control points and the response type of the plant.

h) The sensible and latent gains - radiative and convective - from occupants, lighting and equipment.

i) Condensation

These processes are illustrated in Figure 2-1.

Many of the parameters involved in the analysis vary with time complicating further the simulation. The variation of moisture content in the building materials effects the thermal properties of the materials. The occupancy of the building effects the sensible and latent gains, and also the ventilation rate through windows. The only invariant parameter over time is the geometrical shape of the building. All other parameters are dependent, therefore no one process can be solved independently.

The different analytical methods that have been implemented are discussed in (1) and (2). TAS solves the above system by a response factor type method.

### 2.3 Use of Dynamic Simulation Programs

So far there has been limited usage of these powerful analysis computer programs in practise. There are problems concerning the acceptability of the analysis method. The validity of the various programs is still under constant debate, that is if the modelled building and environment occurred in reality, how close would the predicted results be to the actual results (3).

In addition there are more practical reasons preventing their usage generally, concerning the ease of use:

a) The speed of the analysis has been slow, originally batch processing methods were used on mainframe computers although with more processing power now being available on workstation/ desktop computers this problem is rapidly becoming of less importance.

b) A great amount of numerical data has to be manually entered into the program. This is an error-prone, time-consuming process. To input the data to describe the shape of the building has been in particular a major task. A description of the geometry of each surface in the building is required.

c) Often only a particular class of building can be analysed by one system, and there are no systems which could accept every possibly conceived building design.



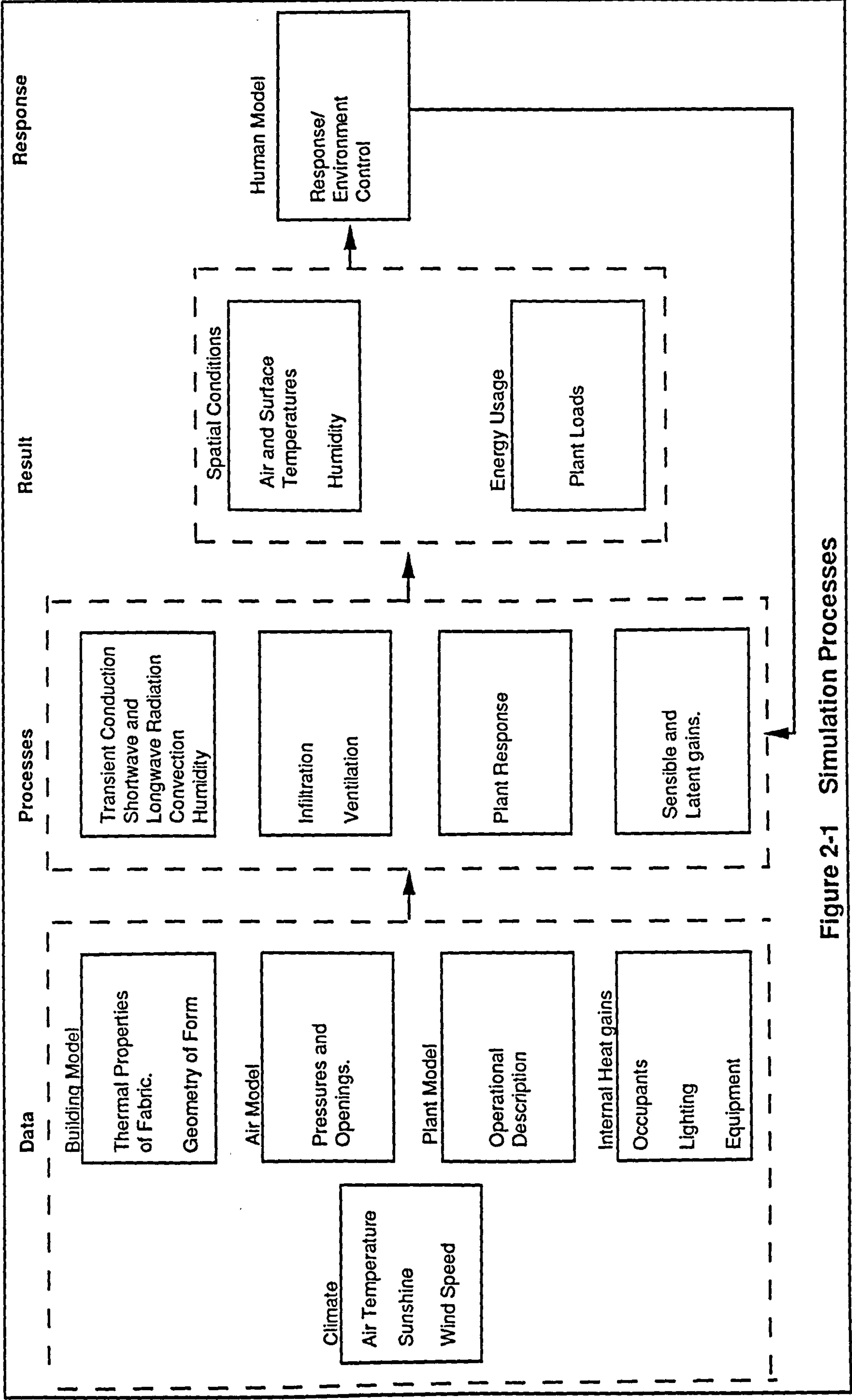


Figure 2-1 Simulation Processes

d) There is also a problem of the interpretation of a model to be analysed into a set of input data. It is often the case that different users will enter different data to describe the same model, and so end up with different results. This is a particular problem with the geometry of the building. Often the geometrical shape of the building has to be approximated for the simulation model, and also to reduce the amount of data which needs to be entered. The building model may be simplified in different ways by different users, and is dependent on the user's judgement (4).

e) To be able to use these simulation programs often requires an expert, there is no simple lead-in for the novice user. For example, often all data describing the problem has to be entered, whereas this could be simplified if some standard default data was available.

## 2.4 Use of Thermal Analysis in Building Design

In addition to the above, there is a further obstacle before the use of these systems can benefit the design of buildings. As described previously, such analysis can influence the following factors in a building design:

- a) The form or shape of the building
- b) The construction materials used
- c) The specification of the heating and/or cooling plant
- d) The control strategy of the plant.

Obviously a) and often b) are defined by the architect at the very start of the building design process.

However the use of any computerised analysis is still often only used by the engineer at the detailed design stages of a building, after the architect has finalised the form and major details of the design, to document the thermal aspects. It is the responsibility of the engineer to make the building design 'work' in these aspects.

This implies that energy efficiency is paid little attention at the early design stages, but it is the decisions made at these stages which have the biggest impact on the building cost and construction, and the ongoing running and maintenance costs. This has been demonstrated in experimental housing projects where energy efficiency has been the most important factor in the design. These designs have been highly effective, although other design factors may have been ignored.

To benefit from the functionality of these systems to influence the design of buildings, the analysis needs to be performed in an iterative manner throughout the design of the building, trying a number of solutions and continuously refining the solution.

## 2.5 A Computerised Design Tool for Thermal Analysis

From the above, it can be seen that there are many aspects in which the form and usage of the current thermal analysis programs could be improved. The necessary characteristics for an ideal design tool are discussed in (5). These are summarised as follows :

- a) Ease of use: an interactive user interface, with prompts and error checks on input data is desirable.
- b) Reproducibility: to ensure that the analysis of a building design by different engineers will produce the same results, there must be little opportunity for input errors or ambiguity in the required data.
- c) Default data: The amount of input data to be input can be reduced by defaulting values whenever possible, allowing the user to override them as necessary, i.e. as he becomes more experienced, or as a design becomes more detailed.
- d) User environments: both architects and engineers may need to perform some analysis, therefore an ideal user interface would adapt to the particular type of user, with emphasis on different input data. e.g. an architect may want to refer to construction materials by name only, whereas an engineer may want to specify a material in terms of its thermal characteristic i.e. U-values etc.

These characteristics can apply to any method of thermal analysis, i.e. both dynamic simulation and steady state. As the same building design may need to be analysed by different methods, this indicates an additional requirement, which is that the same input data is often required for all types of analysis. Therefore, input data should be transferable between analysis methods.

A design tool is required, which can be applied at the early conceptual stages of design, and then throughout the design process to detailed design. At the early stages of design, simple analysis is required which could possibly be carried out by the architect, and which requires little input data, producing quick approximate results. Later analysis on the more developed design will require more detailed input data, will take longer to compute, producing more accurate results.

The input data required for thermal analysis can be categorised as follows :

- a) Shape and form of the building.
- b) Thermal attributes of the constructions of the building fabric.
- c) Internal environment of the building at specific times: occupancy, equipment etc.



d) External environment of the building: weather, orientation of building, and latitude.

e) Plant specifications

The input data which is the most tedious to enter and most prone to error is that describing the geometry of the building. This data is central to a building model, where the other input data can be considered to be attributes associated with it, and therefore reference it. For example the thermal attributes of the constructions are associated with the walls, floors, roofs, windows etc. This implies that particular emphasis should be put on modelling the building form, the way it is input and how it is represented.

The application of an interactive graphical user interface to create the shape and form of the building, with the facilities to allocate attributes to the displayed graphical entities, would provide some of the necessary characteristics of the ideal design tool.

## 2.6 Input of Building Geometry

At present engineers usually take drawings produced by architects, and by hand take measurements from the drawings to produce the necessary data for an energy analysis. They usually have to use their professional judgement to approximate the building geometry into the form required for the analysis program. Therefore different engineers produce different data, which is usually a simplified form of the geometry as most analysis programs cannot cope with complex geometries.

The geometry is represented by sets of numbers, which once entered are very difficult to interpret back to the original drawings and to check.

It is often not clarified to users of these analysis programs exactly what input data is required, and it is often open to interpretation. For example, areas of surfaces are often required to be input, yet user instructions do not give any details of how these surfaces are to be measured. They could be internal, or external areas, or an average of both.

If a building is to be analysed in more than one way, evaluating different design options, then different sets of geometry may need to be input for the same building. However due to the complexity of generating such geometry, often the number of design options are limited.

It is possible that the drawings from which the engineer is taking measurements would have been generated by a CAD system used for architectural design and draughting. Therefore a computerised building model already exists in some form, yet is not being used.

## 2.7 Requirements for interactive graphical interface

From the above the main requirements can be summarised for an interactive graphical interface:

- a) Graphical input to create building model, taking measurements directly from drawings, if available.
- b) If an architectural CAD model exists, then access this to create the thermal building model.
- c) Visualisation of the three dimensional building model, so that the model can be checked and the building design can be communicated to other members of a design team.
- d) Interactive modifications, to allow the refinement of a building model throughout the design process.
- e) Allow the same building model to generate different analysis data for different design options.
- f) Allow the true geometry of the building to be modelled, such that any simplification necessary for analysis can be made automatically when generating the data for analysis.
- g) Interactive assignment of attributes to graphical entities, and display of these assignments for checking.

There has been some previous research in the development of an interface specifically for building energy analysis as reported in references (6) and (7). These have mainly concentrated on c), allowing simple geometric shapes to be created and displayed.

## 2.8 Other Building Design Processes

So far, thermal analysis has been discussed in isolation from the other design processes of a building. Some of the data above is not specific to just thermal analysis. For lighting analysis, the shape of rooms and positioning of windows is required, together with weather data, and the orientation of the building. Obviously additional glazing data would be required to specify the windows etc.

Further environmental analysis: air movement to determine a comfort criteria, also requires the shape of spaces. (This also requires the results of the thermal analysis to provide the boundary conditions for the air movement analysis).

Solar analysis to evaluate the effect of the sun on a building during different seasons of the year, and throughout the day, is a very important related analysis. Shading evaluations are necessary in building design for the aesthetic layout of buildings and in urban design. Calculations for the amount of solar radiation on exposed surfaces of a building are necessary for

energy analysis, and in particular passive solar designs. Also the projection of direct sunlight through windows into a building is similarly of interest. This analysis requires a fairly detailed three dimensional model of the building, including the geometry of its surrounding buildings and features. A graphical representation including shadow projections is necessary for evaluation. These requirements are discussed further in (8).

Therefore, the requirements detailed above for an interactive graphical interface are also applicable to other building design processes.

## 2.9 Building Modeller

The emphasis from these requirements therefore becomes the interactive creation of a geometric model which represents a building design, which can be accessed primarily for thermal analysis, but is also relevant to other analyses. The evolution of the model throughout the design process is also important and how it could relate to the models generated for the architectural design on which it is dependent.

The required configuration is illustrated in Figure 2-2. The main components of the required building modelling system are summarised as:

- a) Geometric modelling method of a building.
- b) Interactive graphical input and modification.
- c) Access of CAD data to create building model.
- d) Visualisation of 3D model.
- e) Assignment of attributes to geometrical entities, as required for analysis.
- f) Generation of data for analyses.



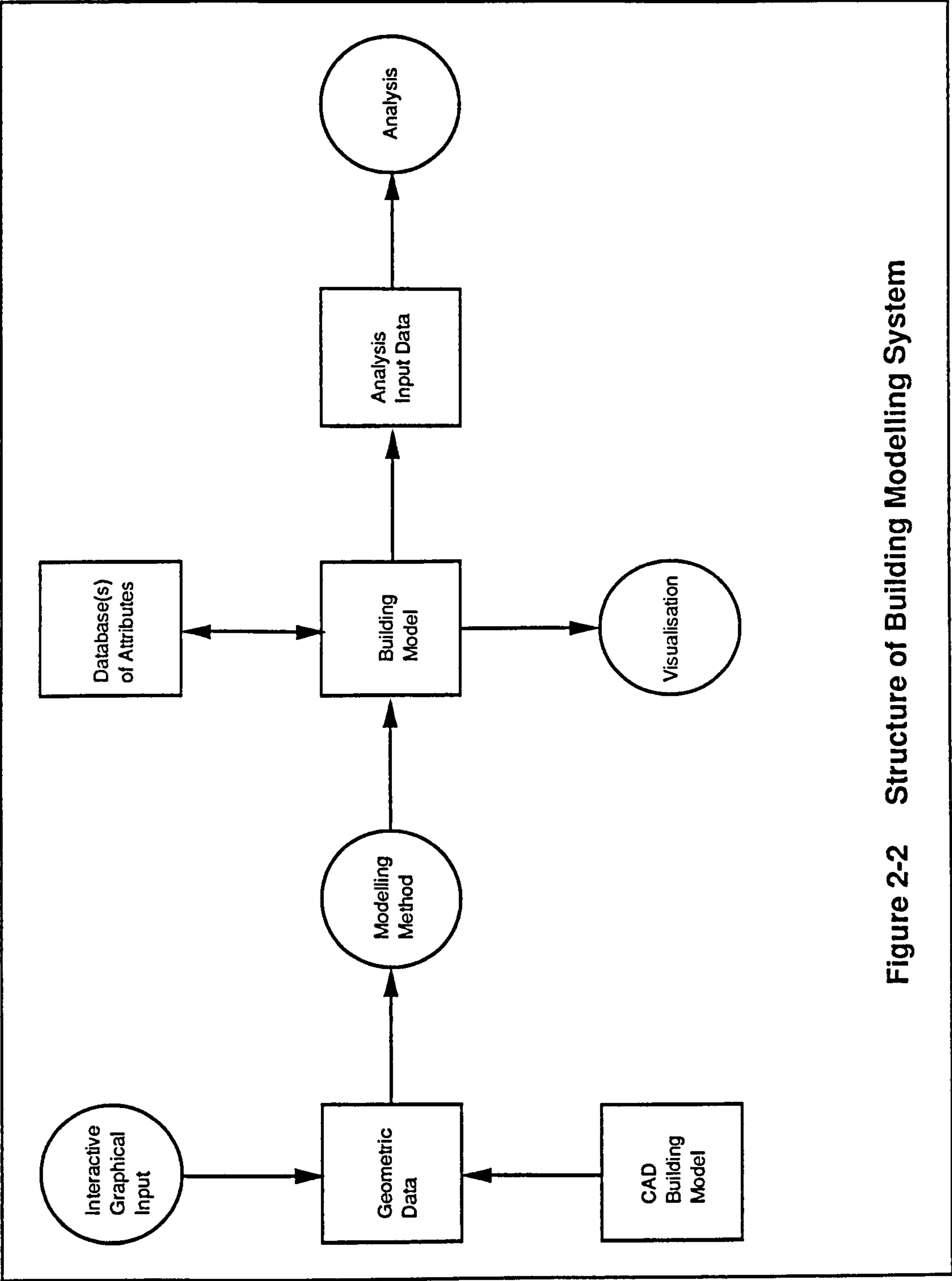


Figure 2-2 Structure of Building Modelling System



### 3. INPUT DATA TO THERMAL ANALYSIS

In order to investigate the form of the geometric model required for the thermal analysis of a building design, the data as required for TAS is now described (9). As stated above, the data required for an analysis can be categorised, the data of particular interest here is:

- a) the data describing the shape and form of the building.
- b) the attributes directly associated with entities of the building shape:
  - (i) the thermal attributes of the building fabric
  - (ii) the internal environment of the building.

#### 3.1 Shape and form of the building

For the purpose of thermal simulation, the building must be divided into zones. A zone is defined as a region in which the air temperature is assumed to be uniform. A zone may be a single space or a collection of spaces.

Each zone is described in terms of its boundary surfaces. For each surface enclosing the zone the following data is required:

- a) Orientation
- b) Slope
- c) Area
- d) Shading Type.
- e) Building Element Number
- f) Building Feature Type
- g) Linking Zone

Of these data items, a) to d) provide the required geometrical properties of the surface. The Building Element Number, item e), indicates the thermal attributes of the surface and is discussed in 3.2. The Building Feature Type and Linking Zone items are dependent on the location and type of the surface, specifying particular attributes required for the analysis of the zone.

##### 3.1.1 Orientation

To simplify the geometrical description of each zone, the surfaces are defined by reference to a cuboid, where each surface is assumed to be aligned to one of the faces of the cuboid.

The aim of this geometrical description is to define the orientation of the surfaces in relation to the compass direction north. Therefore, rather than having to give the orientation of each individual surface relative to true north, two relative orientations can be provided (see figure 3-1):

- a) Building Orientation: defines a building frame of reference aligned with the main axis of the building, and is given as a clockwise rotation from true north.
- b) Zone orientation: for any zone, that is not aligned with the building axis, a further zone rotation can be defined, relative to the building rotation.

The rotation of a surface is given relative to the zone's rotation.

As indicated above, the orientation is specified as being aligned to a face of the cuboid, where these faces are labelled 'North', 'South', 'East' and 'West'. The horizontal faces are labelled 'Ceiling' and 'Floor' appropriately.

### 3.1.2 Slope

Surfaces oriented according to north, south, east or west can also be inclined. A slope in the range 0 to 180 degrees indicates a surface tilted about its lower edge. A vertical surface has a slope of 90 degrees. A surface inclined towards the building e.g. a pitched roof, has a slope in the range 0 to 90. Although occurring more infrequently, a surface inclined outwards from the building has a slope in the range 90 to 180.

### 3.1.3 Area

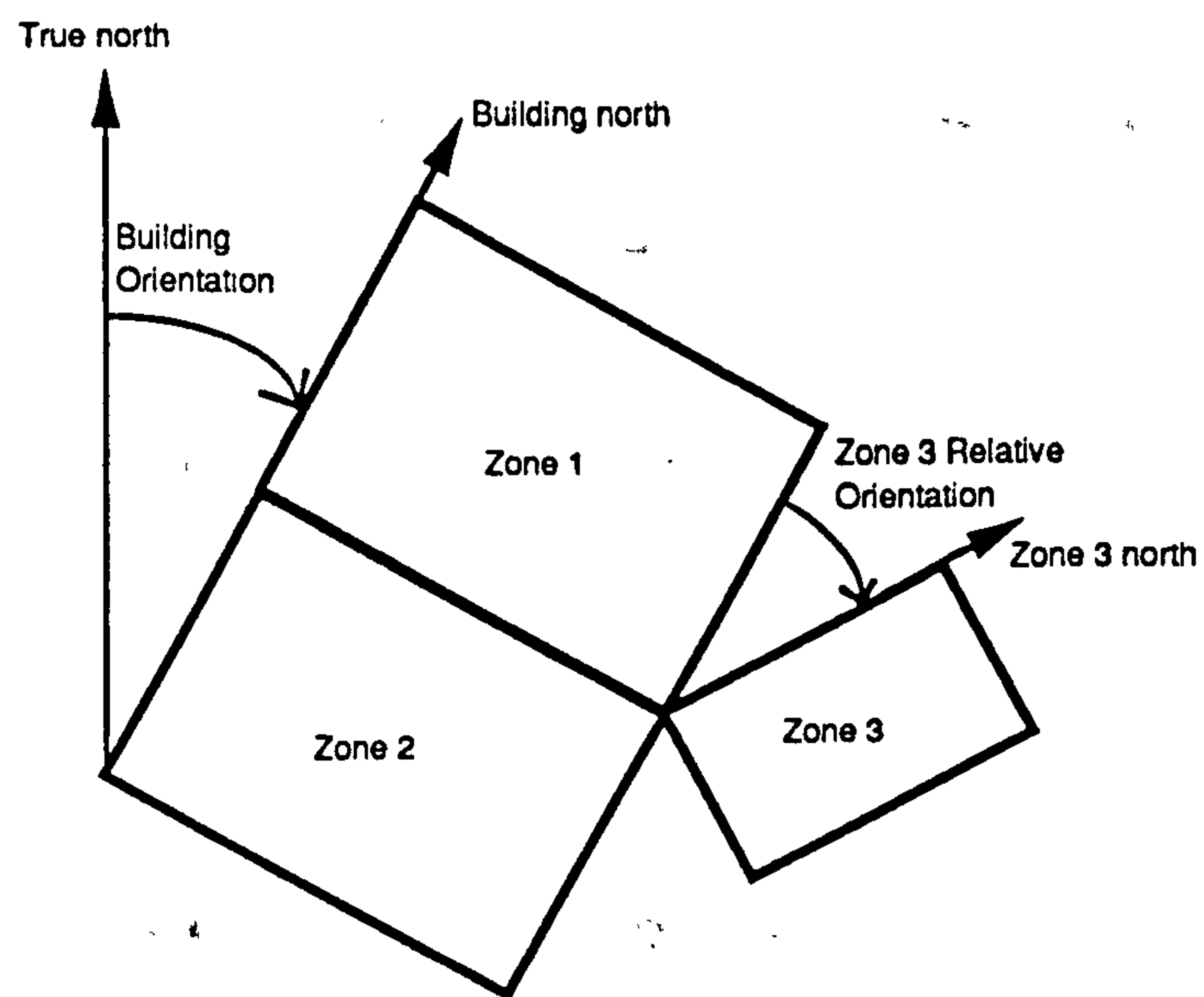
The area of the surface. Surfaces with similar attributes, that is all other surface data is identical, may be combined together into one surface definition, then their areas are totalled. For example, windows facing the same aspect, of the same glazing and shading type, may be defined as one surface of a zone, and their glazed areas are added together.

### 3.1.4 Shading Type

A 'feature shading' can be defined and a surface then references it by this shading type number. This is usually used for windows, where there may be many windows of the same type within a building, with identical recess dimensions.

The 'feature shading' is defined in terms of left and right fins and a overhang. See figure 3-2. The required dimensions are:

- a width of shaded surface
- b height of shaded surface
- c depth of the right fin (right as viewed from outside)
- d depth of the left fin
- e depth of the overhang



**Figure 3-1 Building Orientation & Zone Relative Orientation**

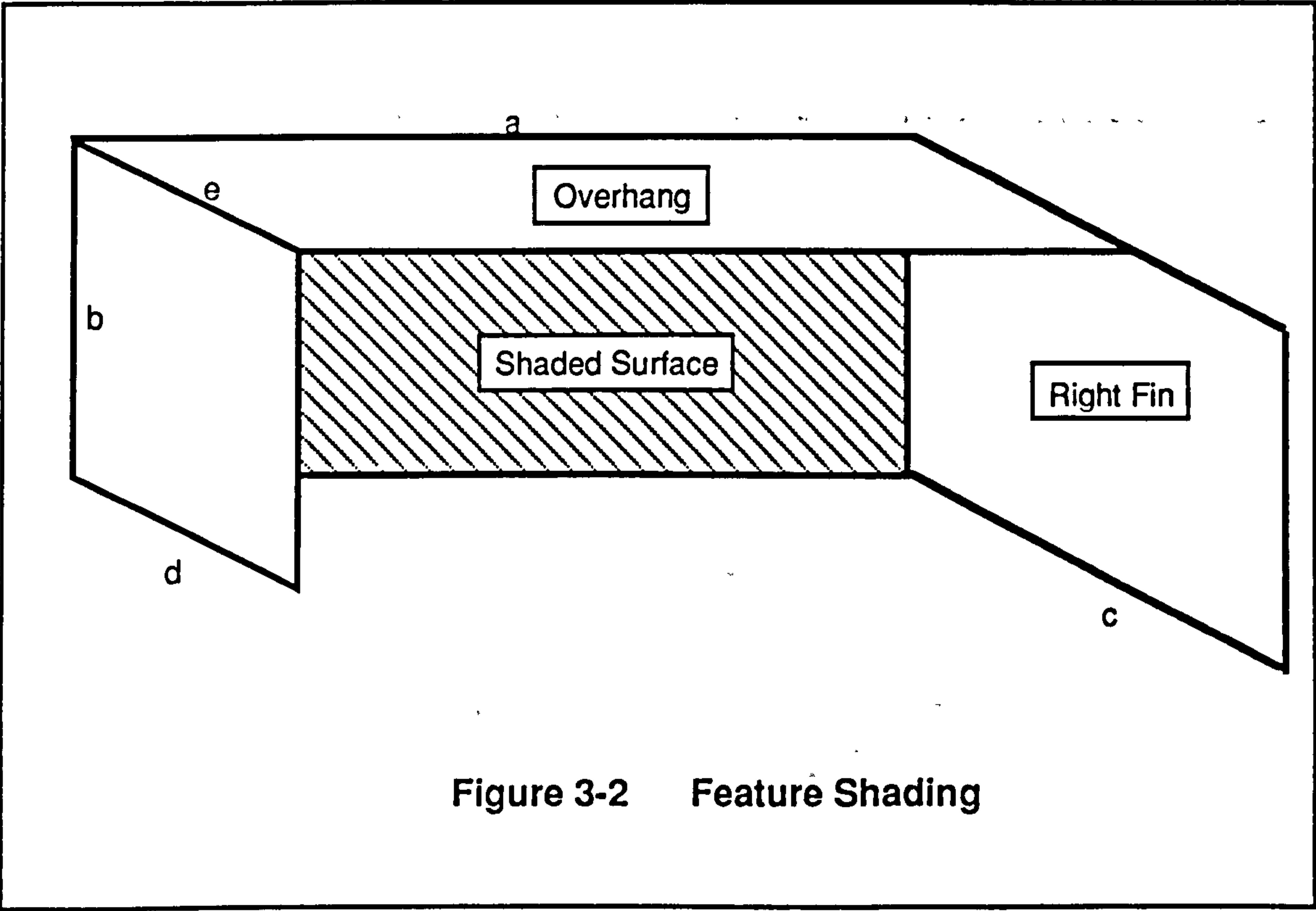


Figure 3-2 Feature Shading



This description defines the relative sizes of the fins and overhang to the shaded surface. When referenced as a shading type by a surface, the width and height of the surface do not need to be a and b above, but should be in the same ratio. The other dimensions, c, d and e, are then adjusted accordingly.

### 3.1.5 Building Feature Type and Linking Zone

There are 5 categories of building feature type:

- a) Exposed: opaque surface, exposed to the external climate e.g. roof.
- b) Transparent: transparent or translucent surface, exposed to the external climate.
- c) Ground Floor: surface in contact with the ground.
- d) Dynamic: surface which connects to another zone. For this type, a 'linking zone' number must also be given, specifying the zone to which the surface connects.
- e) Adiabatic: internal surface which connects to another space with the same environmental conditions as the current zone i.e. no transfer of heat is modelled. There are two uses of adiabatic surfaces. Firstly, the connecting space has not been modelled by a zone. This may occur when a large building is being analysed, and it is known that adjacent spaces are similar. The analysis is therefore simplified. For example, only one floor of a multi-storey building needs to be analysed when all floors are similar. It is not necessary to analyse the heat flow through the ceiling or floor surfaces, and they will be categorised as adiabatic.

The second use of the adiabatic type is for surfaces internal to a zone, i.e. internal partitions. It may be desirable to include these so that their mass is taken into account. In this case, a surface is included for each side of the partition.

To summarise, categories a), b) and c) are used for surfaces forming the external envelope of the building. Categories d) and e) are used for internal surfaces, and whether a surface is adiabatic or dynamic depends on any adjacent zone.

### 3.1.6 Zone Length, Height and Width

The surfaces are described with reference to a cuboid. The dimensions of the cuboid are required for the calculation of the radiation characteristics of the zone interior. For zones that are not cuboid in shape, the length and width are set such that the resulting floor area and plan perimeter are equal to the true values. The height is set such that the volume calculated from the height, width and length is equal to the true volume of the zone.

Figure 3-3 shows a building consisting of 2 zones, and the resulting zone surface data for one zone is shown in figure 3-4.

### 3.2 Thermal attributes of the constructions

To specify the material composition of each of the zone surfaces, two general databases are referenced: Constructions and Materials. These are then referred to by a table of Building Elements created for each building model.

#### 3.2.1 Building Elements

Each surface description of a zone includes a Building Element number. This number refers to a list of the construction types used in the current building. A building element name is assigned to each number for identification e.g. external wall, internal partition. The construction type refers to an entry in a Constructions database and is referenced by a code name. Figure 3-5 shows an example of building elements.

#### 3.2.2 Construction Database

This database stores construction definitions for walls, floors, ceilings, roofs, windows etc, which are created from layers of materials stored in the materials database. An entry is identified by a code. A full name is also entered in the database.

Each layer of material in the construction is specified by its width and a material name. The layers are ordered from inside to outside for external constructions. Internal constructions, e.g. internal partitions are often symmetric in their layers. However, this is not always the case, and in the zone surface data, the building element number is negative if the construction is reversed relative to that surface. Therefore, for each dynamic surface there is a similar surface in terms of area in the specified linking zone, but with opposite orientation and negative Building Element number.

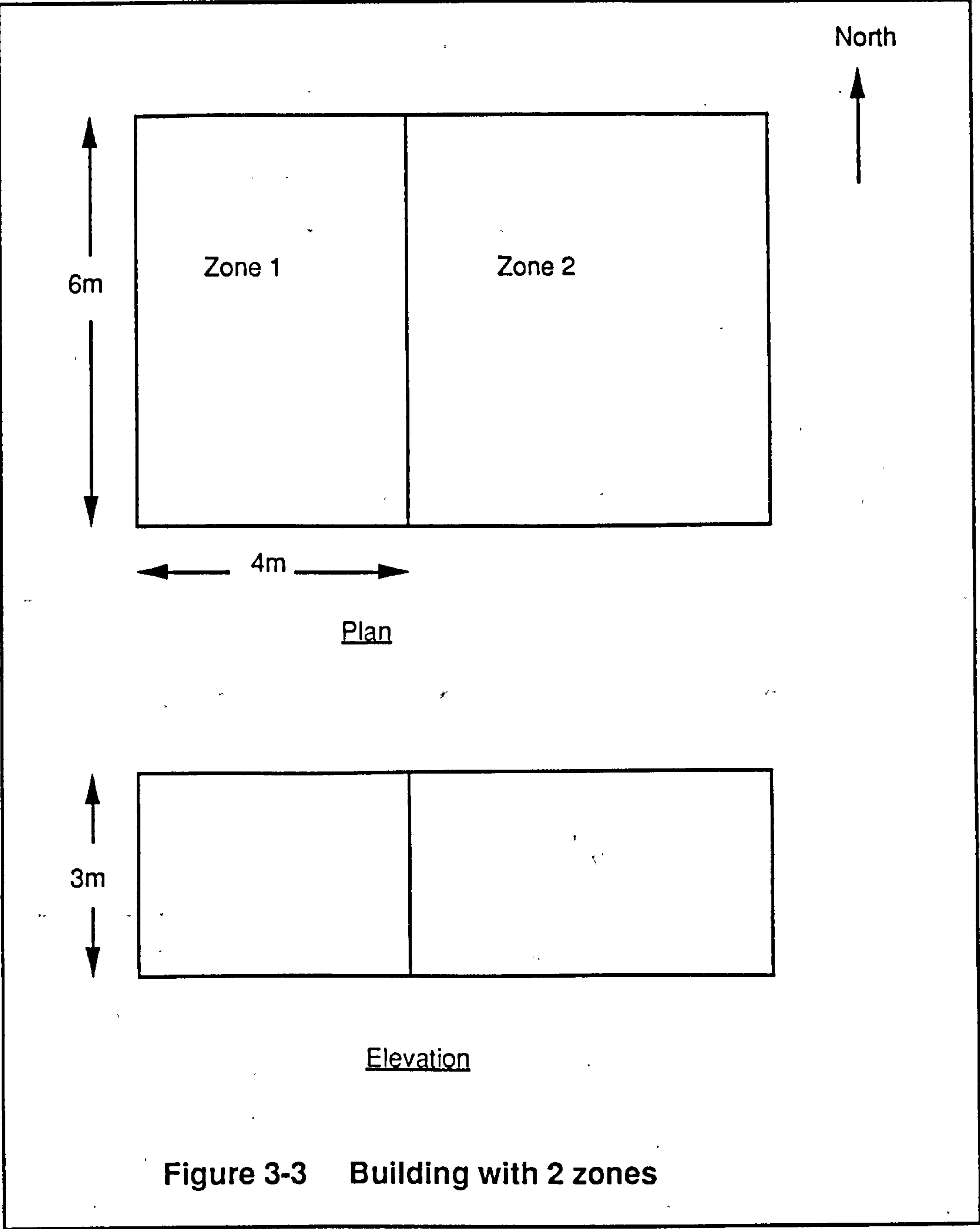
Figure 3-6 shows the specification of a construction.

Where a construction has no appreciable thermal mass, it does not need to be described according to its layers, instead a conductance only can be defined.

For a transparent or translucent construction, a solar transmission coefficient and solar absorption coefficient are set.

#### 3.2.3 Materials Database

The materials database stores data for the thermal properties of building materials. The data is supplied by the Building Services Research and Information Association. Each entry is identified by a code name, and contains the data items shown in table 3-1.



SURFACE GEOMETRY ZONE 1

Volume (m3)	Floor area (m2)	Relative orientation (degrees)	Multiplied by
72.0	24.0	0.0	1

Surface number	Building feature	Orient-ation from N (degrees)	Slope (degrees)	Building Element number	Area (m2)	Linking Zone number	Shading feature number
1	Gnd Floor	Floor	0.0	1	24.0	0	0
2	Exposed	Ceiling	0.0	2	24.0	0	0
3	Exposed	North	90.0	4	12.0	0	0
4	Exposed	West	90.0	4	18.0	0	0
5	Exposed	South	90.0	4	12.0	0	0
6	Dynamic	East	90.0	3	18.0	2	0

Figure 3-4 Surface data for one zone

BUILDING ELEMENT TABLE

Building Element Number	C - Code	Construction Name	Building Element Name
1	df/1	FLOOR	Ground Floor
2	dc/1	ROOF	Ceiling
3	diw/1	INT WALL	Internal Wall
4	dew/1	EXTERNAL WALL	External Wall

Figure 3-5 Example of Building Elements



CONSTRUCTIONS DATABASE

C - Code:	dew/1	Construction name: EXTERNAL WALL
-----------	-------	----------------------------------

OPAQUE CONSTRUCTION

External Solar Absorptance	Internal Solar Absorptance	External Emissivity	Internal Emissivity	Conductance (W/m2 C)	Time constant (hours)
0.5	1.000	0.900	0.900	0.487	7.0

Layer Number	M-Code	Width (mm)	Conductivity (W/m C)	Density (kg/m3)	Specific Heat (J/kg C)	Convection Coefficient (W/m2 C)	Vapour Diffusion Factor
Inside	amls/20	0.10	999.999	0.0	0.0	-	0.010
2	amlblock/1	150.00	0.317	1040.0	1050.0	-	14.800
3	amlins/1	50.00	0.040	12.0	833.0	-	2.880
4	amlair/1	100.00	0.556	0.0	0.0	-	1.000
5	amlbrick/1	105.00	0.700	1700.0	800.0	-	8.000
6	amls/10	0.10	999.999	0.0	0.0	-	0.010
7							
8							

Figure 3-6 Example of a Construction



Data Item	Description
Conductivity	Thermal conductivity of the material.
Density	Density of the material.
Specific Heat	Specific heat capacity of the material.
Diffusion Resistance	Specific moisture diffusion resistance of the material.
Resistance	Only for a special class of items representing air-gaps, it is the thermal resistance of the air-gap. Conductivity, density, and specific heat attributes are not used in this case.

Table 3-1: Entry in Materials Database

### 3.3 Internal environment of the building

The Internal Conditions are specified for each zone, that is a time-dependent description is given of environmental control, internal heat gains and fresh air infiltration and ventilation.

Each group of data describes one day, and for each zone a different set of data is used to describe a weekday, a Saturday and a Sunday. Therefore when the simulation is performed over a given period specified by a range in days, the assumption is made that the first day of the year, day 1, is a Monday, and the appropriate Internal Conditions data can then be accessed for each day for every zone.

The day is divided into periods of time for which the fresh air infiltration and ventilation, and internal gains are specified. The day can be divided up into a maximum of eight periods, each defined by an hour range.

#### 3.3.1 Internal Heat Gains

Sensible heat gains and latent heat gains (moisture gains) are specified for the zone dependent on the occupants and equipment within the zone. The heat gain in the zone due to artificial lighting is also specified.

### 3.3.2 Environmental Control

The environmental parameters, temperature and humidity can be specified in terms of a range of acceptability, with the aim that this is maintained by any plant.

### 3.3.3 Air Infiltration and Ventilation

The fresh air infiltration and ventilation into the zone is specified as air changes per hour. The actual air flow is then dependent on the zone volume. Ventilation is defined as air entering the zone via a mechanical ventilation system.

Air movement between zones may also be specified. It is given as the mass air flow into each zone.

### 3.4 External environment of the building

TAS performs a simulation of the specified building over a period of time dependent on the external environment. This is defined by a set of weather data, dependent on the location of the building. A climate database contains weather data for different regions of the UK, and also for some locations around the world.

Each set of weather data contains the hourly values for a year of a number of weather parameters. These data items are shown in table 3-2.

Data Item	Description
Global radiation:	Total solar radiation intensity on a horizontal plane.
Diffuse radiation:	Diffuse sky radiation intensity on a horizontal plane.
Sunshine duration:	A factor varying from 0 for overcast conditions to 1 for a clear sky. It is used to estimate long-wave sky radiation.
Dry Bulb Temperature	
Wind Speed	
Relative Humidity	

Table 3-2: Hourly Weather Data

The weather data also indicates the latitude of its location, which is required for calculating the direction of the sun at any hour throughout the year.

### 3.5 Plant specifications

The plant is specified on a zone by zone basis within the Internal Conditions data, again as a time dependent description. Up to four plant operating periods can be specified within a day, for each of which the heating and cooling capacities available for the zone are given.

The type of temperature control of the plant available within the zone is also specified. This determines how the plant responds to fluctuations in temperature. Also whether a plant responds to changes in external temperature, that is the range of external temperature within which the plant is operational.

Further plant control parameters are specified to determine how the plant operates outside of normal occupancy periods, to ensure there is a preheat period before occupation starts and to protect against frost.

### 3.6 Summary of Geometric Model

TAS requires the user to approximate a building design into a simple set of rectangular zones. The required data describing the geometry of a building to be analysed consists of the following data items :

- a) Building
- b) Zones
- c) Surfaces

#### 3.6.1 Zones and Spaces

A zone is one or more spaces, where the Internal Conditions as defined for the zone are the same for each space. The volume of the zone is the sum of the volumes of each space within the zone.

A zone number can therefore be regarded as an attribute of a space, so describing the internal environment of the space, in terms of environmental control and occupancy and associated plant.

The zonal arrangement of the building defines which spaces are to be analysed. Each space within a building for a particular analysis is either:

- a) within a numbered zone, to be included within the analysis, -or
- b) not within a zone, not included in the analysis,

That is the allocation of a zone number to a space determines whether it is to be included in the analysis or not. A building can be represented, therefore, as a set of spaces, where each space may or may not be allocated a zone number.



### 3.6.2 Surfaces and Constructions

The surfaces of each zone are derived by considering the walls, floors, ceilings, roofs i.e. the constructions of the building adjacent to each space of the zone.

If a wall is internal, that is partitions two spaces, two surfaces will be derived from it, one on each side of the wall. This is assuming that on each side of the wall there is a zoned space. When the zones of these two spaces differ, the surfaces are typed as 'dynamic', one surface definition allocated to each zone. The linking zone number is provided in the surface data. When the wall partitions two spaces of the same zone, the surfaces are typed as 'adiabatic', and two surfaces referring to the same wall appear in one zone's data. This applies similarly for spaces arranged vertically, where a floor to ceiling construction is considered.

Therefore, for the surface data to be consistent, there always should be two surfaces derived from any internal wall or floor to ceiling construction, when both adjoining spaces are zoned. The two surfaces must be of equal area and opposite orientation. If only one space is zoned, only one surface is defined, of type 'adiabatic'. It is assumed that there is no heat flow through the wall, between the zoned and unzoned spaces. When a wall or roof is exposed, or a floor is adjacent to the ground, only one surface is derived of type 'exposed' or 'ground floor'.

The building element to which a surface refers defines the construction type of the wall, floor etc. As previously described a construction type is made up of layers of materials. The layers are not necessarily symmetric, this is more common for an exposed construction, for example the layers defining a wall from the internal plasterwork through to exposed brick, but is feasible for internal walls also. Therefore to indicate the orientation of the construction type in relation to the surface, the Building Element number is given as positive when the first layer defined for the construction is adjacent to the zone of the surface, and negative when the last layer is adjacent to the zone.

To summarise, each building construction generates potentially two surfaces in the zone data, viewed from opposite directions. The construction type is an attribute of a building construction and is associated by a building element number, where its orientation relative to the construction must also be defined.

### 3.6.3 Zonal Arrangement

The input data defines the building for one analysis only, that is for one zonal arrangement. Often different zonal arrangements may need to be considered when designing a building, or during the design process from sketch to detailed design, when more zones may need to be analysed, by further dividing zones. Each zonal arrangement requires different data to be input to the analysis, yet it is derived from the same building geometry: from the walls, floors, ceilings, roof that represent the building's unvarying shape.

There may be some data which can be reused, but often a completely new set of data is required because of the way the input data is organised on a zone by zone basis.

Included in the input data are items that are directly dependent on the zonal arrangement i.e. distinguishing between adiabatic or dynamic type for an internal construction, in the latter case a linking zone number must be provided. To generate this it must be possible to identify the adjacent space, to determine whether or not it is zoned, and if so its zone number.

#### 3.6.4 Requirements of Geometric Model

TAS requires very little geometric data to be input, compared to some simulation models. The simulation program ESP (Environmental Systems Performance) allows a zone to be defined by full geometric data in the form of vertex co-ordinates and surfaces referring to the vertices (10). The ability to deal with complex building geometries also varies substantially. However the common definition is that a building is divided into zones, and the surface geometry for each zone is described in some form or another. Therefore it should be possible to define a building model that will satisfy the requirements of other thermal analyses other than TAS, and can be accessed to generate the input data for that analysis.

The above indicates that a building can be modelled as two dependent sets: of spaces and of constructions, with the requirement that the following are derived from the model:

- a) the constructions surrounding each space, that is forming the boundary of each space.
- b) the spaces adjacent to each construction.
- c) the constructions forming the external envelope of the building.
- d) the geometry of each construction, so that further geometrical properties can be derived i.e. area.
- e) further, from a) and d), geometrical properties of a space i.e. volume.

By definition, the geometric model must be an unambiguous representation of a building, such that all data generated from the model for input into the analysis can be ensured to be consistent.



In addition, it is necessary to associate attributes with the building model:

a) A zone number is assigned to a space of a building, for two purposes:

i) to associate a set of Internal Conditions data

ii) to indicate that the space is to be included in a particular analysis

b) A construction type is assigned to each construction: wall, floor, ceiling, roof, window of a building to define the thermal attributes of the building fabric.



#### 4. CURRENT CAD SYSTEMS AND MODELLING TECHNIQUES

The geometric model which needs to be created for a thermal analysis to be possible has now been described. As the geometry of a building can be modelled by existing CAD systems used in architectural design and draughting, the current status of such CAD systems are now considered, in terms of the modelling techniques used, and also their applicability to the design process for thermal analysis. This will give an indication of whether :

- a) The required geometric model for thermal analysis is created by current geometric modelling techniques.
- b) A model created by an existing CAD system could be accessed to provide at least some of the data necessary for thermal analysis.

As discussed previously, the geometry of a building is central to many analyses, and therefore ideally there should be one CAD system from which all the necessary geometry for the various analyses could be accessed. As the building model was developed through each stage of the design process, so the requirements of geometrical data would also change to match the differing analysis types.

The different forms of modelling used in CAD systems are summarised below, particularly with reference to architectural applications.

##### 4.1 2D Draughting System

These were the first type of CAD systems to be produced and mimic the way a traditional draughtsman works. Graphical data, consisting of lines, arcs and text are created within the computer, which can then be plotted any number of times.

The ability to manipulate the data within the computer, using powerful editing facilities to delete, copy and move data, and so produce revisions of drawings, is the main advantage of these systems. They are an automation of the draughting function i.e. drawings are produced.

The data stored in the computer is a representation of a drawing only, it does not model the object of the drawing, a building, for example. Therefore, there is no co-ordination between different drawings: plans at differing scales, elevations, even though they may be of the same building. The graphics data has no meaning, or interpretation. For example, two lines may represent a wall of a building, but there is no way to access the data to determine this.

##### 4.2 2D Modelling

These systems are now the more common 2D systems. Graphical data is grouped together into components to represent a single physical object for example, a chair, a window. The component can then be placed many times in a 2D plan model, possibly in true 2D co-ordinate space. The component can



be transformed: rotated, mirrored, before positioning, If the component is edited, this effects a global change throughout the model.

Drawings can be created from the 2D model by specifying the area of a model, a plotting scale, and often additional text to be plotted as part of the drawing: a title block.

The components are often categorised for various applications such as structural, electrical, furniture and fittings. These categories can be grouped on 'layers' of a model. So different layer groups can also be defined to be plotted as a drawing.

Libraries of components are usually created, to be used in many different projects, for example a library of office furniture components. Some may be defined 'parametrically', that is a basic shape can be assigned different physical dimensions to produce a family of components.

Attributes are often associated with components, such that some scheduling and costing may be calculated from the model.

With this technique the data has more structure, in that there is a hierarchical structure within the model, and it would be possible to access the data and identify all walls, and possibly obtain the dimensions of the wall: the height could be stored as an attribute. However a different model is still required for each floor plan of a building. There is no co-ordination between floors, for example the location of stairs, and again there is no three dimensional co-ordination to produce elevation drawings.

### 4.3 3D Modelling

To provide co-ordination between different orthographic views, a 3D model of an object must be created.

The first type of 3D models were wireline models, and represented an object by a set of edges in 3D space, an edge was defined by two 3D co-ordinates. Any view of the object created from 3D projections e.g. perspective, axonometric, oblique was then possible.

However it was very quickly realised that these views are often ambiguous. To provide a more realistic picture, a definition of the surfaces of the object is required to hide the edges at the back of the object. Surfaces were then modelled, usually by identifying the edges which bounded the surface. Initially only planar surfaces were modelled, then simple surfaces such as cylinders, spheres, cones, and later more complex sculptured surfaces.

Using the same hierarchical approach as 2D modelling, very complex 3D scenes can be created by positioning a 3D model of an object, defined in edges and faces, within 3D space, applying rotation and translation transformations.

The main use of such models has been for visualisation, to produce different views of the same model, with increasing realism: shading according to a light source, shadow generation, transparency, reflections. This has many advantages in that information on a 3D object or an assembly of objects can

be conveyed between a design team with much greater understanding than that given by traditional orthographic projections.

#### 4.4 Solid Modelling

The 3D surface modelling described above can however model unrealisable objects. An object can be modelled by a set of facets, which do not necessarily form the closed boundary of an object. For many engineering applications where analysis is as important as visualisation, a consistent representation of an object is required. The modelling technique known as solid modelling ensures that this is so.

Attributes such as the volume, weight and moments of inertia can be calculated for modelled objects, and the object can also be analysed for the design of the manufacturing process, e.g. tool path generation.

An object can be modelled in a number of ways. A 'boundary representation' records edges and facets of an object as in surface modelling, but the adjacencies between faces are also known. The boundary representation is created in such a way that consistency is ensured. The most common operators used are Euler operations which construct the topology of edges, vertices and faces. These are discussed in more detail later.

A second method of solid modelling is known as 'Constructive Solid Geometry' (CSG). This uses set or boolean operators to manipulate primitive solids for example, blocks, cylinders, cones. The assumption is that if the primitives are consistently defined i.e. solid objects, the operators to combine them will produce a consistent result. Transformations are usually applied to the primitives: scaling, rotation and translation, before they are combined.

A 'CSG' tree is stored within the computer, recording the operations, primitives and transformations used at each step of the construction process. An evaluation of the tree into a surface representation is usually necessary to visualise the object.

A hierarchical assembly of solid objects can be created by applying a 3D transformation to the objects. This is useful for applications such as collision checking in robotics. An unlimited number of realistic 3D images can be created, as for 3D surface modelling.

#### 4.5 Building Modelling

Current CAD systems providing solid modelling are not applicable to the architectural design of buildings. There is no need to create a consistent model of every component part of a building, where the main requirement of 3D modelling of an individual component is for visualisation purposes.

Building modelling has differing requirements throughout the design process, from displaying realistic perspectives of initial sketch design, where a client is to be briefed on a proposed design, to the production of very detailed co-ordinated orthographic drawings which contain construction details for use on site.

Whereas the manufacturing method of an engineering object can be derived directly from the solid model, until robotics are introduced into the construction industry, the design and method of construction of the building is communicated via these drawings.

There is a limited requirement for clash detection between pipes, ducts etc, where solid modelling might be applicable, but this is just one application in the building design process.

A further major difference between mechanical engineering applications, where most solid modelling software is aimed, and building design, is that a building is composed of a very large number of repeated components.

To summarise, the main use of 3D modelling in architectural CAD is for:

- a) Visualisation of 3D model as a realistic perspective picture.
- b) Co-ordination of orthographic drawings.

Unfortunately, the nature of architectural drawing conventions is such that it is very difficult to combine these two applications into one 3D model. This is illustrated very simply by considering the basic architectural elements walls, windows and doors.

A wall can be modelled in 3D as a rectangular block, and this is quite adequate for realistic perspectives. However, when that wall is shown on a plan drawing, it is usually shown in section, using drawing conventions derived from the wall type, such as hatched areas for cavity walls. A solution would be to generate a plan drawing by taking a section through the 3D block, then passing this to a 2D draughting system for embellishment, to add the additional lines and hatching. This is not acceptable as all walls of the same type would need the same additional graphics, and any modification in the 3D model, such as moving a wall, would necessitate the regeneration of all drawings.

Similarly considering windows, a plan drawing shows the cross section of all windows in walls, yet there may not be a height at which a plan section could be generated from the 3D model that shows all windows in cross section. In a ceiling plan, windows do not need to be shown at all.

In a plan drawing a door is shown as open, with the door swing indicated, yet in an elevation it is shown as closed.

Therefore there are conventions in architectural drawings, such that symbols differ between different views. Often these symbols are generated by other designers, that is they are not an integral part of the current building design, for example furniture and fittings symbols represent a suppliers catalogue.

The main co-ordination in orthographic drawings, is in the positioning of symbolic representations, not necessarily in the geometry of the representation itself.



Some integrated CAD systems have been developed, aimed at building design providing the required co-ordination between symbolic views, and true perspective views of a 3D model.

One such system, RUCAPS, is described in (11). This is achieved by referencing instances of positioned components in true 3D space, where the definitions of the components are stored in a library. A component is defined by a set of 2D drawing symbols and a 3D surface model. There is a 2D symbol for each of the required orthographic projections: plan, north, south, east and west elevations and sections. The appropriate symbol is drawn dependent on the required drawing of the assembled model. The 3D model of a component is defined by faces, edges and vertices, and is used to create realistic perspectives.

When a drawing is created, the appropriate component representation is used, dependent on the drawing definition. A drawing symbol may contain text, hatching, dimensions etc, all the functionality usually provided by 2D draughting systems.

A further building modelling system, GABLE, distinguishes between the 3D components forming the building structure, that is walls, windows, floors and ceilings, and the objects: fittings and furniture placed within the building. When positioning walls, the user indicates whether internal or external walls are being defined, and can also create a set of linked walls. The drawing detail, the sectional representation and junctions between walls are automatically generated for working drawings from the 3D building model.

The system also allows specification attributes to be associated with the building structure entities, such as construction materials. This makes it possible to perform a simple steady state thermal analysis on the building model, using the definition of internal and external walls.

The data is organised on a floor by floor basis, so a complete building model consists of a set of floor models. To summarise, therefore, the GABLE database is highly structured in its representation of a building, such that the structural entities can be easily identified and accessed.

#### 4.6 Use of Architectural CAD systems during the design process

Most usage of CAD in architectural design is at the detail design stage, where it is used to finalise layout and then in the production of drawings, that is producing the design documentation. Although there is some use at sketch plan stages, there appear to be prejudices regarding the use of computers by designers at these more creative, earlier stages.

It could also be argued, however, that the CAD tools available are not suitable to early conceptual design. The current CAD systems impose too much structure on the way a building has to be modelled, making it very difficult for a design to be refined as it develops, or to make major design changes easily. For example, walls have to be modelled as an instanced wall component before the type of wall is known, and yet at preliminary design stages, there is more interest in communicating the overall three dimensional shape and spaces of the building.

#### 4.7 Applicability of CAD systems to environmental design

In order to evaluate whether a building modelled by a CAD system, either a 2D or a building modeller, is in the form required for environmental design, the following observations are made:

- a) Buildings tend to be modelled as a set of instanced components, either in 2D or 3D co-ordinate space. With this modelling method, there is no 'room' entity, that is a room or a space is not represented explicitly. Therefore, a model could not be interrogated to find 'all the walls that enclose a space'. In fact the length of an instanced placed wall may border more than one room.
- b) Following on from the above, there is no need to ensure a space is modelled as fully enclosed by the components that surround it, with closed openings, and with a floor and ceiling. If only the layout of rooms is of interest, only the walls and internal fittings may be modelled, with no ceiling component defined.
- c) As there is no concept of a space, neither is there any concept of 'inside' or 'outside' of the building, or spaces that are adjacent to other spaces. There is therefore ambiguity in the building model. The GABLE system distinguishes between external and internal walls, but this is within the users control and is therefore dependent on the user modelling correctly.
- d) Relationships between entities are not necessarily modelled. For example, a window can be placed in a location that is within a wall, and can be seen as such. However, the relationship 'the window is placed within the wall' may not be explicitly recorded in a graphics database.
- e) There is usually more detail in an instanced component definition, than is required for an energy analysis. A window may be modelled including a frame and sill, when all that is required is the basic shape, from which an area of glazing can be derived. It is also possible that there would be more detail in the floor layout, in terms of walls, windows etc. than required. An environmental engineer may use his judgement to decide that certain features would have no significant impact on any energy analysis, and decide they need not be represented in a thermal model of the building. N.B. this is not the same as simplifying the geometry of the building for analysis.
- f) Areas of surfaces are sometimes held as attributes of components, but are not usually derived directly from the geometry of the component. A function is often defined depending on other parameters. For example, an area of a wall is defined as the product of two parameters: its length and height, and is calculated when the wall is instanced. The storage of such attributes as data can depend on the way that a CAD system is used.

In summary, therefore, it appears that current CAD systems used for architectural design do not create a model of a building in a form which could be accessed directly for an energy analysis. The GABLE database provides the most readily accessible data, in that the 3D components representing walls, windows, floors and ceilings can be identified. The geometrical positioning of the structural components, could be used to provide the basic input to create the required model. To make use of models

created by other CAD systems, it would be necessary to constrain the way they are used, i.e. define component naming conventions, or category numbering such that their database could be accessed and all structural components identified.

Previous research development producing an interactive graphics interface in energy or shading analysis, (6) and (7), have created geometric models by combining primitives to represent the zones of a building, as in solid modelling systems. This requires the user of the system to interpret the building form into such a representation. The faces of the primitive volumes then provided the required geometry for analysis. This model represented just one zonal arrangement. Modelling a building in this way does not resemble the way CAD systems for architecture normally operate. It therefore would be difficult to provide functions to modify the model, to add, delete, and move walls.

Ideally it should be possible to derive the required unambiguous 3D model, including relationships and attributes, directly from the architectural design model, and update it as it evolves. To do this with current CAD systems would require an interpretation of the component based data representing walls and windows, possibly floors, ceilings and roofs.





## 5. BOUNDARY REPRESENTATIONS IN SOLID MODELLING

In previous research, solid modelling methods have been used to represent the zones of an analysis, i.e. by using primitive volumes. Also solid modelling was developed so that an unambiguous model of an object is created such that analysis is possible. Therefore, the following are considered:

- a) The domain of objects that can be modelled
- b) The way in which such models are represented and constructed to achieve the well-formedness condition.

### 5.1 Topology of Boundary Representation

Solid modelling is a method of modelling which guarantees to represent an object in a consistent manner. An object can be considered to be a partitioning of points in 3D Euclidean space into:

Points within the object

Points outside of the object

Points within the object can be further divided into:

Points totally interior to the object

Points on the boundary of the object, adjacent to those outside of the object

For a solid model of an object, it must be possible to determine for any 3D point the set to which the point belongs.

A boundary representation of an object defines those points on the boundary of the object. The topological properties of this representation is now summarised. Further detail is given in (12) and (13).

#### 5.1.1 2D Manifold

The bounding set of points can be seen to be a 2 dimensional manifold. A 2D manifold is a space where each point possesses neighbourhoods which are homeomorphic to an open disk.

#### 5.1.2 Connected Manifold

A surface is a 'connected' 2D manifold, that is there is a continuous path between any two points of the manifold.

### 5.1.3 Bounded Surface

A 'bounded' surface is one which can be contained within a finite sphere: all distances between points on its surface are finite. This excludes the infinite plane and cylinder.

### 5.1.4 Closed and Orientable Surfaces

Solid modelling systems are concerned with modelling only realisable objects. There are two further properties a bounding surface must possess for this to be ensured. The first of these is that a surface must be 'closed'. This eliminates objects with isolated edges and lamina, that is homogeneous continuity is ensured.

The second property to ensure a realisable object is that the surface must be orientable. The orientability condition known as Mobius' Law eliminates one-sided objects and self-intersecting objects.

Therefore, to summarise, the boundary of a well-formed solid object, i.e. one that is realisable in 3D Euclidean space, is a closed, orientable 2D manifold.

## 5.2 Euler-Poincare Formula

A property of geometric figures known as the Euler-Poincare formula was identified at an early stage in the development of object modelling by Baumgart (14) as being relevant to ensuring well-formed models.

Euler initially proved a property of geometric figures, which states that for any closed convex polyhedron, e.g. a cube, where  $V$  is the number of vertices,  $E$  the number of edges, and  $F$  the number of faces, then

$$V - E + F = 2$$

This property only applies to those polyhedra which are homeomorphic to a sphere, that is for a closed 2-dimensional manifold with a genus of zero. A topological model which illustrates the property of a higher genus is that of a sphere with handles. A surface which is homeomorphic to a sphere with  $g$  handles is of genus  $g$ , e.g. a torus is of genus one.

Poincare generalised this property for  $n$ -dimensional manifolds. The resulting Euler-Poincare formula, simplified for a 2D manifold therefore states :

$$V - E + F = 2 - 2G$$

where  $G$  is the genus of the surface.

Therefore by ensuring that this formula is valid for any object created by a modelling system, and in the process of its creation, some degree of well-formedness is guaranteed.

### 5.3 Embedded Graphs

The Euler-Poincare formula is stated in terms of the vertices, edges and faces of an embedded graph in an orientable closed surface. An embedded graph is one which can be drawn on a surface such that no two edges meet except at a vertex.

In a boundary representation, an object is defined in terms of the vertices, edges and faces of an embedded graph in the surface of the object. A set of constraints derived from those defining the closure and orientability of a surface ensure that this representation is consistent:

- a) Faces intersect only at shared vertices or edges.
- b) The faces around each vertex form a single circuit.
- c) A path must exist between any faces of the representation, where the path is formed by crossing shared edges.
- d) The vertex pair of each edge is ordered once in each direction, for adjacent faces, when the vertices around each face are ordered consistently.

A further extension has been made to the Euler-Poincare formula (15) to allow the representation of objects by disjoint graphs. A face is then bounded by one or more 'loops' of edges. Also, more than one bounding surface may need to be modelled at any time, so a further extension to the formula allows for multiple bodies or 'shells'. Where  $H$  is the number of hole loops and  $S$  the number of shells, the formula is stated as :

$$V - E + F - H = 2S - 2G$$

### 5.4 Graph-based Data Structures

Data structures have been developed which represent an embedded graph, that is the data elements: vertex, edge and face are explicitly defined in terms of their adjacency relationships and geometrical attributes. The most well known of these is the winged-edge data structure (14).

From a data structure it must be possible to derive any topological relation between the entities. For example, the edges which form the boundary of a face, the faces adjacent to a vertex. There are nine possible adjacency relations between vertices, faces and edges, the result of which is an ordered or unordered list of entities. (This assumes a face is not multiply connected i.e. it is bounded by a single loop of edges and vertices). See table 5-1.



	Face	Edge	Vertex
Face	Circular ordered list of faces sharing edges.	Circular ordered list of edges forming boundary of face.	Circular ordered list of vertices forming boundary of face.
Edge	Two faces on either side of the edge.	Edges sharing either of the vertices.	Two vertices at either end of the edge.
Vertex	Circular ordered list of faces with the vertex on their boundary.	Circular ordered list of edges connected to a vertex.	Vertices sharing an edge.

Table 5-1 Adjacency Relationships between Topological Entities

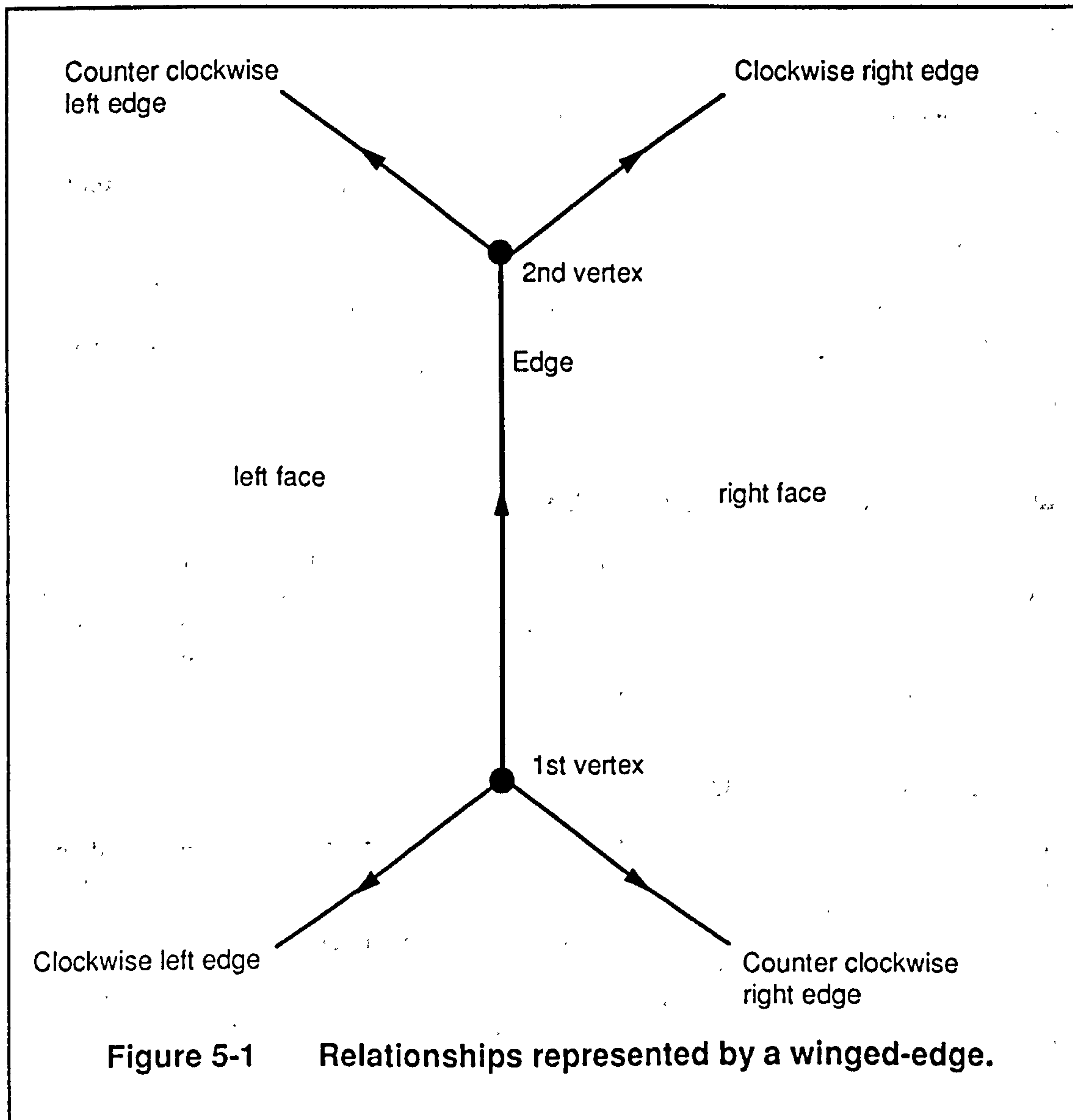
In the winged-edge data structure, the main topological relationships between vertices, edges and faces are expressed with reference to each edge. It specifies the Edge-Edge, Edge-Face and Edge-Vertex data explicitly, while all other relations are derived. Figure 5-1 illustrates the relationships which are represented by an edge entity. This data structure has the advantage of having fixed length data records, rather than storing lists of data which may vary in length, for example, the edges bounding a face.

Other data structures have been developed (16), which are variations on the winged-edge data structure, and have been investigated for their sufficiency in representing an embedded graph, and their efficiency in terms of storage and access requirements.

### 5.5 Definition of data records

In order to define the data records for a winged-edge structure, the assumption is made that each edge, face, vertex is uniquely identified. This means that each item is represented by a group of data, and each group of data can be referenced by a pointer. The form of the pointer will depend on the computer language of the implementation. For example, in Pascal and C, it would be a pointer to a record data type, whereas in Fortran it may be an integer which is an index to an array of data. The indicated record or data array would then contain the group of data for that item, mainly pointers to other items.

The derivation of such data records is now considered.



### 5.5.1 Face loops

As discussed above, objects can be represented by disjoint graphs, so introducing a new data item: a face loop, where the boundary of the face is represented by more than one loop of edges. Therefore, instead of referencing a left and right face, a winged-edge must reference a left and right loop, and a face is defined as a list of loops.

### 5.5.2 Hierarchical Structure of Object

Access of a data structure representing an object is often top-down, such that all shells of an object are required in turn, and then all faces of each shell. The following, therefore, all need to be available as lists: shells of an object, faces of a shell and loops of a face. To enable this access, downward pointers and sideways pointers are usually implemented: from the object to a shell, and from each shell to the next shell so that a list can be formed; similarly for shells and faces, faces and loops. A pointer is also implemented from a loop to an edge, and a vertex to an edge.

### 5.5.3 Special case of single vertex loop

When forming a winged edge data structure using Euler operators, described below, a special case can arise such that a loop consists of a single vertex. This is also possible when some non-planar geometries are represented. Therefore, a loop may reference a vertex instead of an edge and similarly a vertex may need to reference a loop.

### 5.5.4 Wire edges and closed edges

There are two special cases of edges which can occur in an embedded graph. A wire edge is one which has the same face i.e. loop on both sides. This usually occurs during the creation of the data structure (see Euler Operations below). A closed edge has the same vertex at either end. This can occur with certain geometrical definitions (see 5.7 below). These edges can be represented in the standard winged-edge representation, but with the same pointer, either vertex or loop, appearing twice.

### 5.5.5 Additional Backward and Sideway pointers

Dependent on the application, other pointers are sometimes implemented. So that it can be determined the object to which an item belongs, backward pointers are sometimes implemented from each item : shell, face, edge, vertex to the object. Backward pointers may also be implemented at each level of the hierarchy: shell to object, face to shell, loop to face. In order to create a list quickly of all the edges or all of the vertices of an object, without having to trace through the hierarchical structure, and eliminate duplication, a sideways pointer is implemented from each edge to the next edge for the object, and similarly for a vertex.



### 5.5.6 Description of Data Records

Below, in tables 5-2 to 5-7, the data record for each entity is described, with possible field names for each data item. The optional pointers described in 5.5.5 are omitted from the descriptions.

#### EDGE

Field Name	Description
pvert	1st vertex of edge - 'Previous vertex'
nvert	2nd vertex of edge - 'Next vertex'
lloop	Loop of left face adjacent to edge
rloop	Loop of right face adjacent to edge
elcc	Edge counter-clockwise from edge in left loop
elcw	Edge clockwise from edge in left loop
ercc	Edge counter-clockwise from edge in right loop
ercw	Edge clockwise from edge in right loop

Table 5-2: Edge Data Record

#### OBJECT

Field Name	Description
shell	1st shell of object - usually outer shell

Table 5-3: Object Data Record

#### SHELL

Field Name	Description
nextshell	Next shell in list for object. Circular linked list.
face	1st face of shell

Table 5-4: Shell Data Record

**FACE**

Field Name	Description
nextface	Next face in list for shell. Circular linked list
loop	1st loop of face - usually peripheral loop

Table 5-5: Face Data Record**LOOP**

Field Name	Description
nextloop	Next loop in list for face. Circular linked list.
edge	An edge of the loop, or NULL pointer if there is no edge.
vertex	Single vertex of loop, if loop has no edges.

Table 5-6: Loop Data Record**VERTEX**

Field Name	Description
edge	An edge attached to the vertex, or NULL pointer if there is no edge.
loop	Loop of single vertex, if vertex has no attached edges.

Table 5-7: Vertex Data Record

### 5.6 Access of topological relationships

In an implementation of the winged-edge structure it is necessary to provide a set of access functions, that create the required list of entities for each of the nine possible adjacency relationships. Substituting 'loop' for 'face' to allow for faces with multiple loops, with the assumption that the face is then implied by the loop, only two relationships can be derived directly from the data structure :

- i) Edge - Loop: left and right loop of edge.
- ii) Edge - Vertex: previous and next vertex of edge.

For the others, there are four lists to be derived:

- i) Edges of loop: for loop - edge.
- ii) Vertices of loop: for loop - vertex.
- iii) Loops of vertex: for vertex - loop.
- iv) Edges of vertex: for vertex - edge.

The remaining are 'self relations' and have been defined in relation to edges and are in fact a combination of the above relations:

- i) loop - loop = opposite loop of each edge in 'edges of loop' list
- ii) vertex - vertex = opposite vertex of each edge in 'edges of vertex' list
- iii) edge - edge = 'edges of vertex' list for pvert  
+ 'edges of vertex' list for nvert

In order to create the required lists, it is necessary to search through the winged-edge data, from edge to edge, identifying the required related items. For these operations it is necessary to derive one entity, an edge, loop or vertex, given two other entities, one of which is an edge, and the other is referred to in winged-edge data of the edge. The following set of functions are therefore suggested (17) as being of use for data access.

- i) ccwvc:  
Gives the next edge clockwise around a vertex from an edge :  
if vertex is nvert of edge then next edge is elcc  
else vertex is pvert of edge then next edge is ercc
- ii) ccwel:  
Gives the next edge clockwise from an edge in a loop :  
if loop is lloop of edge then next edge is elcw  
if loop is rloop of edge then next edge is rcw
- iii) lcwev:



Gives the loop such that an edge is clockwise from a vertex in the loop :

```

      if    vertex is pvert of edge    then loop is rloop
      if    vertex is nvert of edge    then loop is lloop

```

iv) vcwel:

Gives the vertex of an edge clockwise in a loop :

```

      if    loop is lloop of edge      then vertex is pvert
      if    loop is rloop of edge      then vertex is nvert

```

A similar set of functions can be defined to access in a counter-clockwise manner.

The above assumes that no wire or closed edges are present in the structure. The access needs to be modified to allow for the fact that these may be present.

### 5.7 Definition of Geometry

So far the representation of just the topology of an embedded graph has been described. The geometry of the model also needs to be specified and be consistent with the topology, in particular to ensure the orientability of the model. This association of geometry, such that the topology constrains the geometry is discussed in (15).

Each topological entity needs to be geometrically defined:

- a) Face: by the surface in which it lies c.g. plane, cylinder
- b) Edge: by the curve in which it lies c.g. straight line, circle
- c) Vertex: by the co-ordinates of the point.

As stated in 5.3, orientability is ensured if faces only intersect at edges and vertices. Therefore, the geometry must be consistent also: the intersection of the surfaces of 2 adjacent faces must be along the curve specified for the edge, the intersection of the curves of two or more edges must be at the point of the vertex to which each edge is connected.

Dependent on the implementation, i.e. the types of geometry to be represented, additional fields need to be added to the data structures above for each entity type, either including the geometrical definition explicitly, or by including a pointer to a geometrical entity. As indicated above 'closed' edges may be used to represent some geometrical forms, for example, the circular edges of a cylindrical tube. A single vertex loop may be used to represent the apex of a solid cone.

### 5.8 Euler Operators

To ensure that the embedded graph data represents a consistent object, operators known as Euler operators have been developed (14,15) to create and

modify the topology of a model. These operators ensure that the Euler-Poincare formula is always satisfied.

An Euler operation is a subtraction or addition in the numbers of vertices, edges, faces, hole loops, shells and genus such that the formula is still valid.

The derivation and choice of Euler operators is discussed more fully in (17) and (18). The most common Euler operations are shown in figure 5-2. The inverse operators are also usually required.

The operators have been investigated (19) in terms of soundness: any embedded graph created by them has a physical realisation, and in terms of completeness: the surface of any solid object can be modelled by a finite sequence of them.

## 5.9 Implementation of Euler Operators

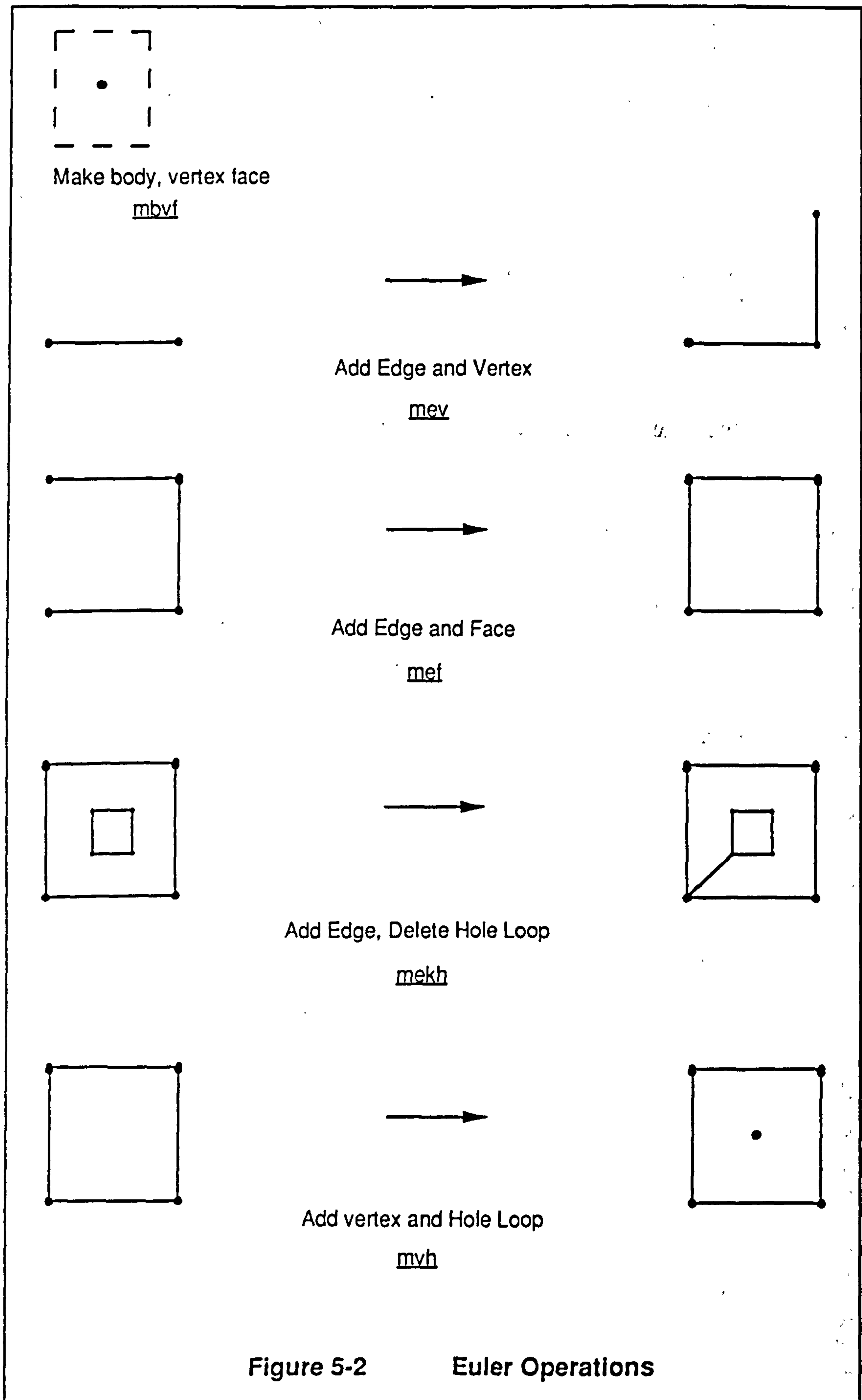
Within a modelling system, to ensure that the embedded graph data structure is kept consistent, all additions and modifications to the topological entities should be through a set of functions that perform the Euler operations. The functions need to be specified from the application environment from which they are called. The specification is given in terms of input parameters, stored parameters, functionality, assumptions made as to the validity of input parameters, and any conventions adopted. The following points relate to the specification of these functions.

### 5.9.1 Geometry

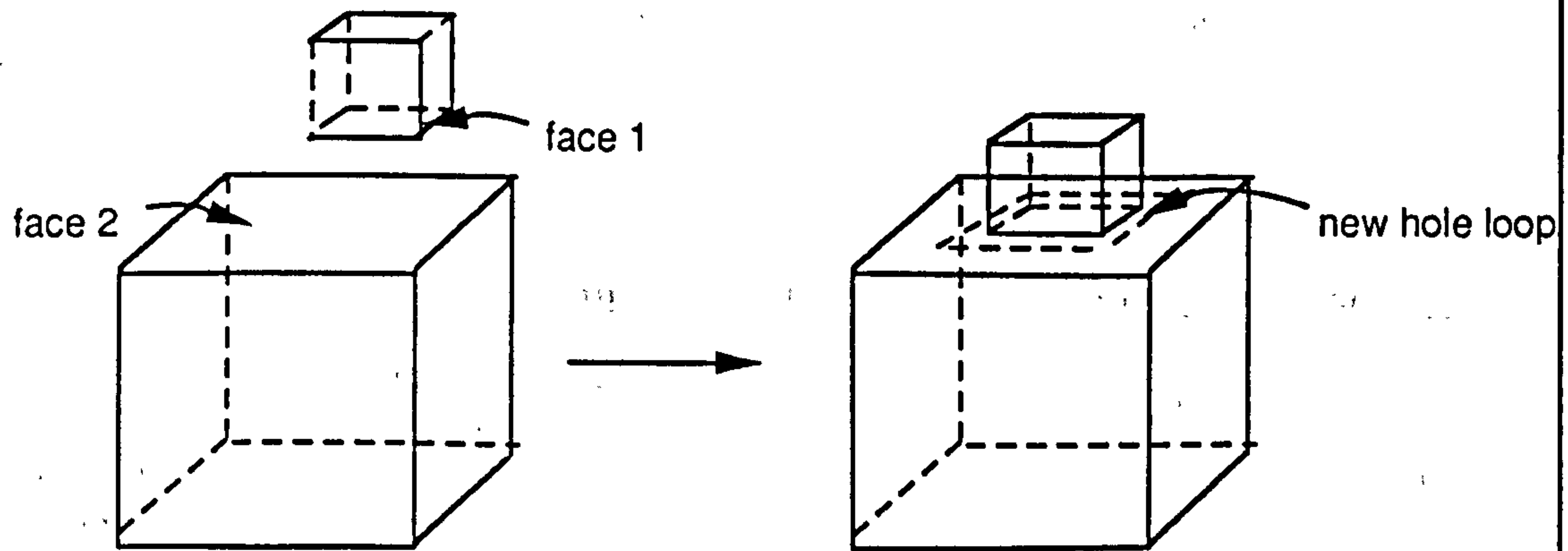
The geometry of each entity may be specified at the time of creation, when the function is called to perform an Euler operation, or may be 'attached' to the entity subsequently. The geometry is usually regarded as an attribute of the topology. It is very difficult to maintain a topologically and geometrically correct model during the creation of topology, so the assumption is usually made that the input parameters are 'correct' in specifying a consistent object and no checks are made on the geometry within a function to ensure, for example, that the intersection of the curve of 2 edges is at the point of their vertex or that when connecting two vertices with a new edge, no other edges are crossed.

### 5.9.2 Input Parameters

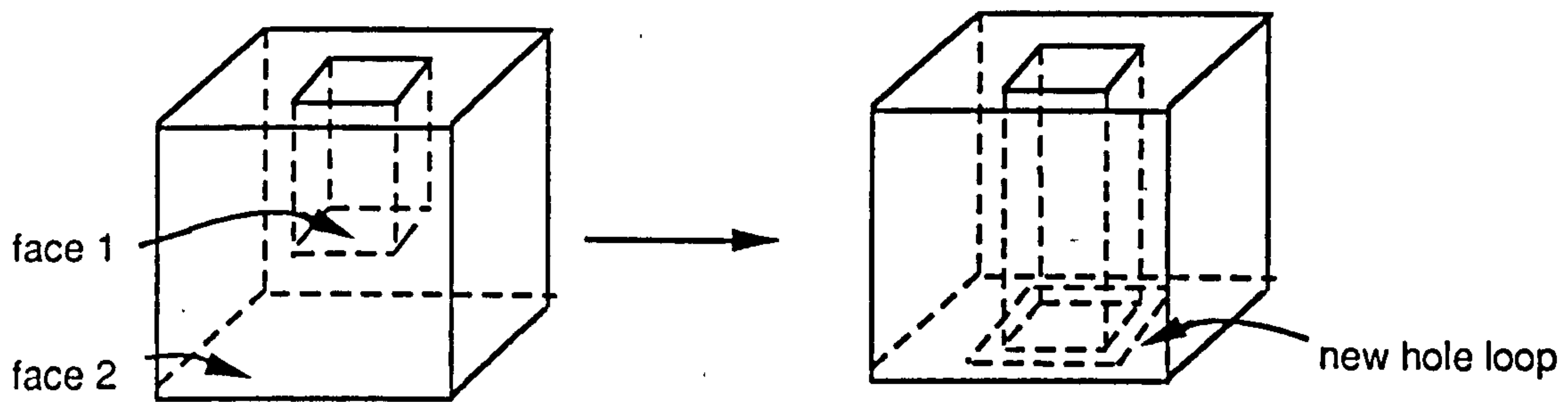
When creating topology, the input parameters to each function must sufficiently specify the connectivity of new entities, especially as no geometrical properties can be assumed to be available. Edges are mainly used to construct the topology, for which it is necessary to indicate the vertex to which an edge is being connected to. Also the edge that will be clockwise to the new edge around the vertex is given to indicate how it is to be connected. If closed edges are possible within the topology, then the loop into which a new edge is to be added must also be given. This is explained more fully in (18).







Make hole, kill face, body  
mhhfb



Make hole, genus, kill face  
mhgkf

Figure 5-2 (Continued) Euler Operations

The operation to add an edge and create a face has the effect of dividing an existing face into two. As there is no geometrical checking, it is possible that there are hole loops of the original face that should now be within the new face. It is therefore necessary to specify, as part of this operation, any hole loops to be moved.

### 5.9.3 Returned Parameters

An indication of the topological entities that have been created is usually given, such that they can be subsequently identified. This requires, therefore, that each entity has a unique identification, available at least during the application when created, possibly as an identifier into data stored in external files.

### 5.9.4 Operation performed by a function

Composite functions can be defined, for example to add an edge between two vertices, whereby within the function itself the appropriate Euler operation is performed: either add edge and make a face, or add edge and kill a hole dependent on whether the two end vertices are in the same or different loops of a face. Alternatively a function can be defined for each operation if it is assumed that the calling application can generate the correct operation.

Conventions usually need to be defined to specify a function completely. For example when dividing a face into two with an add edge and face operation, it is necessary to specify which side of the edge the new face lies; when splitting an edge, it is necessary to specify to which vertex of the original edge the new edge is attached.

The specification of a set of functions performing Euler Operations has been documented in (20).

## 5.10 Components of modelling system.

The three major components of a modelling system for a boundary representation have been described above. To summarise, these are:

- a) Data structure: such that the required domain of objects can be represented. The winged-edge data structure has been described in detail.
- b) Data structure access: such that all required adjacency relationships can be determined.
- c) Creation of topology: to ensure the representation is consistent and well-formed, the specification of a set of functions to provide Euler operations has been considered.

Often higher level operations are used within a modelling system, specified in terms of Euler operations, such as the definition of simple volumetric primitives, profile sweep operations. These operations usually ensure the consistency of the geometrical properties of the model.





## 6. BUILDING MODEL REPRESENTATION

Having investigated the way that a solid model can be represented by its boundary, and how such a representation is created and accessed, the required representation of a building is now considered.

### 6.1 Spaces and Constructions

As discussed in Chapter 3, the geometry and topology of the required building model can be defined in terms of spaces and constructions, with any specified zonal arrangement generating the surface data on a zone by zone basis ready for analysis. A building can be modelled therefore by a set of spaces, whose adjacencies are of importance, and a set of construction elements, where these form the boundaries of the spaces.

The attributes of a space are its internal conditions data and are associated by a zone number. The attribute of a construction is its construction type, defined by layers of materials, associated by a building element number.

The requirement is to create a building model with attributes, such that all geometric data: zone volume, surface areas, surface orientations, and all topological data: surface types, linking zones, can be derived directly from the model.

### 6.2 Comparison of Model with Boundary Representation

An individual space has a boundary which is a closed surface, like a solid object. Similarly the external envelope of the entire building is a closed surface. However all the spaces need to be considered as a dependent related set.

As points could be partitioned for a solid model representation, here a point can also be considered to have unique set membership. A point can be:

External to the Building

On the external envelope of the building

Within the building, within one space

Within the building, on the boundary of at least 2 spaces.

There exist, therefore, conditions of well-formedness. It is feasible to assume that a 3D model of a building can be represented by the set of points on the boundary of the spaces, where a point may lie on the boundary of 2 or more spaces and be internal to the building, or lie on the boundary of 1 or more spaces on the external envelope of the building. See figure 6-1. Point A is internal and on the boundary of two spaces. Point B is external and on the boundary of one space.

Considering the point A in 3D space, it is not locally homeomorphic to a disk, and therefore the boundary is a non-manifold representation.

### 6.3 Representation of Space by Embedded Graph

As discussed in Chapter 5, the bounding surface of a solid object can be represented by the faces, vertices and edges of an embedded graph, where a geometrical definition is associated with each topological entity i.e. a surface definition for a face, a curve for an edge and a point for a vertex. The bounding surface of an individual space can also be represented by an embedded graph.

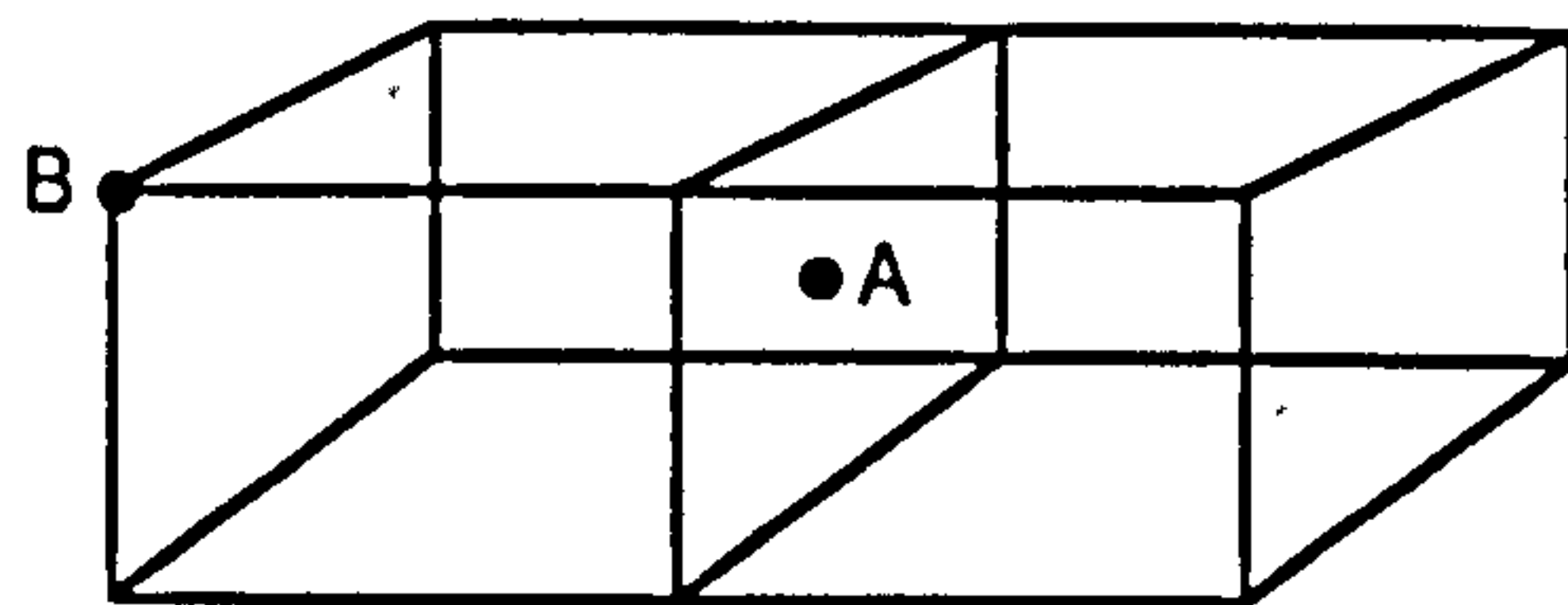
However, if every space was represented in this way, there would exist coincident faces for all neighbouring spaces. Some edges and vertices of the faces would also be co-incident. See figure 6-2. If this duplication could be recorded, however, it should be possible to model the complete boundary of the spaces in a consistent manner.

A building construction can be considered to be a lamina with 2 faces, geometrically identical, but with opposite orientation, i.e. opposite normals. Each face either forms part of the boundary of one space, or is an external face and forms part of the boundary of the external envelope. The lamina is conceptually of zero thickness. Although the thickness of a construction is considered within the analysis when calculating heat flow, it can be considered to be an attribute of the construction type and not relevant to the topology of the building model.

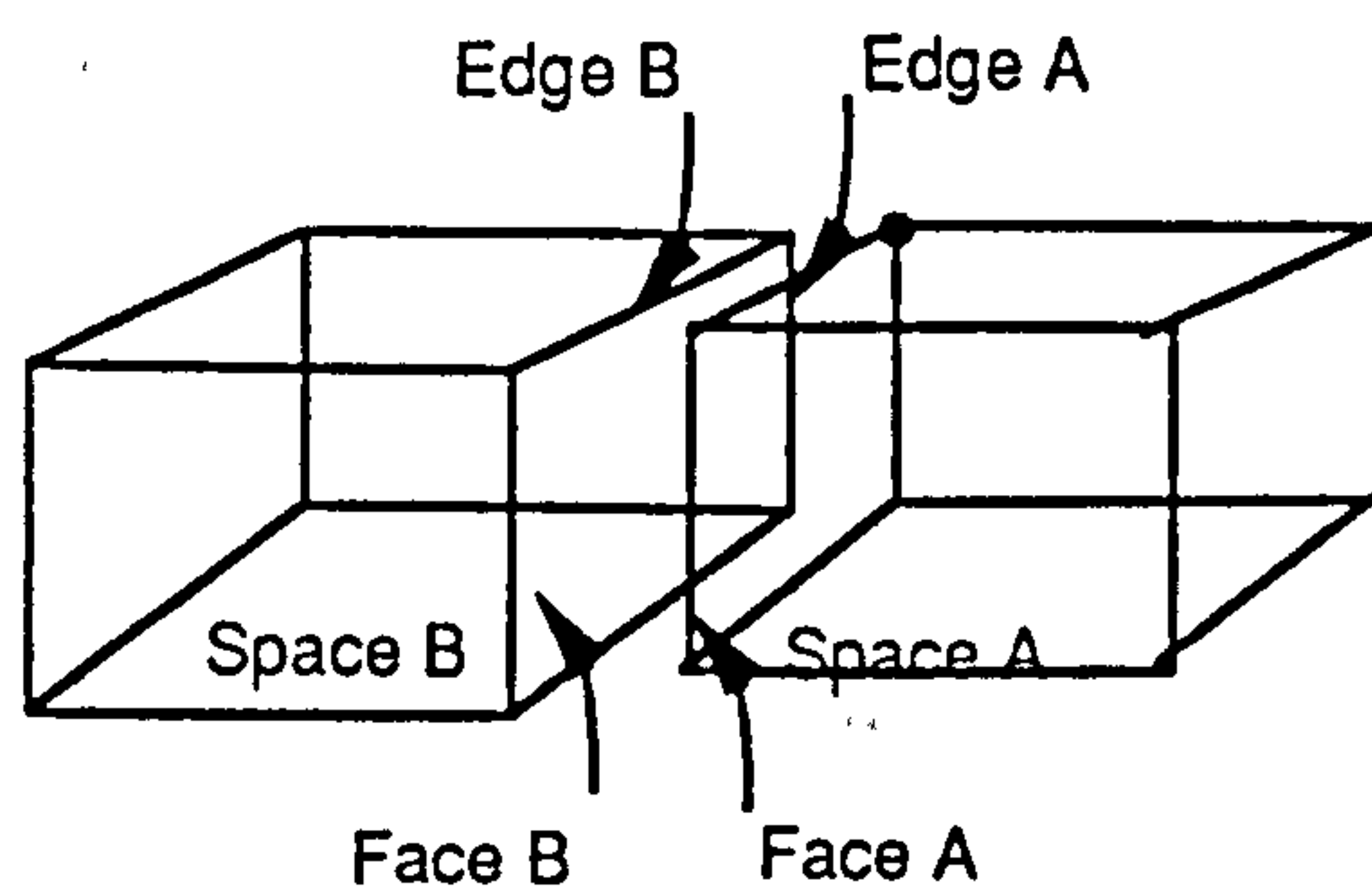
A construction, therefore, can be defined to represent the duplication of two faces in different embedded graphs of two adjacent spaces. The two faces must be 'geometrically identical', that is the faces have to lie in the same geometric surface and be bounded by a 'geometrically identical' set of edges and vertices. Conversely, this means that where the boundary representation of adjacent spaces could be defined by faces that lie in the same surface, but with different boundaries, as in figure 6-2, if the regions enclosed overlap, the faces must be divided into the overlapping and non-overlapping regions, each resultant face in a different construction, so defining constructions as specified above. See figure 6-3. Face B has been divided into 2 faces: B<sub>1</sub> and B<sub>2</sub>, where a construction is formed from faces B<sub>1</sub> and A.

Similarly an entity can be defined which groups the coincident and geometrically identical edges of the faces together, hereafter called a 'multi-edge', and one which groups the vertices together, called a 'node'. For two edges to be 'geometrically identical' they must lie in the same curve, and have coincident endpoints. In figure 6-3, vertex A and vertex B are vertices of the same node, and edge B<sub>1</sub> and edge A are edges of the same multi-edge.

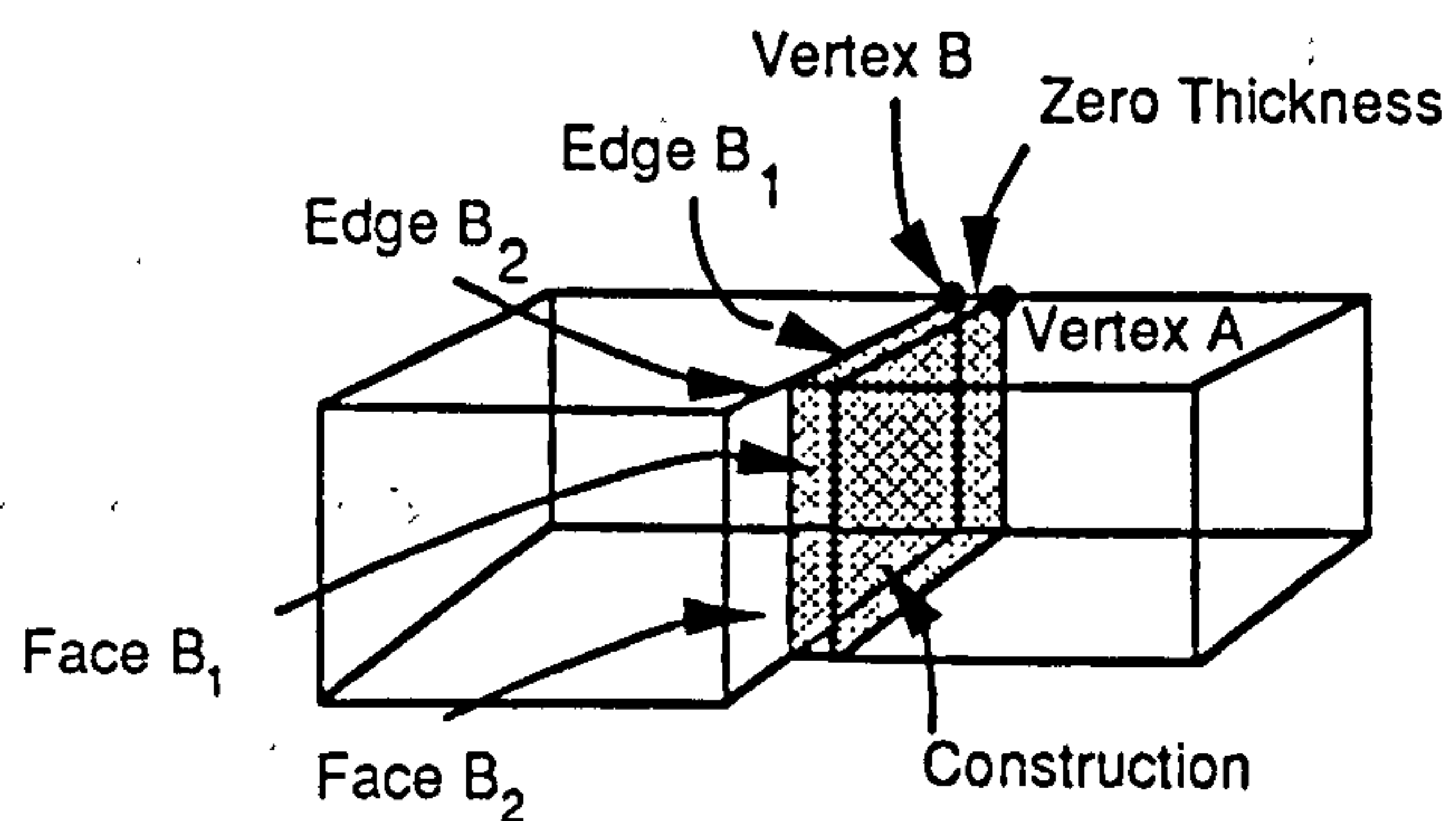
Again, as faces must be divided into non-overlapping regions, so there cannot be two edges coincident along the same curve, with different endpoints. The segments of the edges have to be derived to form separate multi-edges. Edge B in figure 6-2 is shown divided into edge B<sub>1</sub> and edge B<sub>2</sub> in figure 6-3.



**Figure 6-1 Points on space boundaries**



**Figure 6-2 Representation of adjacent spaces by different embedded graphs**



**Figure 6-3 Construction represented by two co-incident faces**

In summary, therefore, it should be possible to represent a building by a set of embedded graphs, one for each space, and by constructions, nodes and multi-edges, to define the relationships between the embedded graphs.

A construction is defined as:

2 'geometrically identical' faces, where each face forms part of the boundary representation of a different space.  
See figure 6-4.

A multi-edge is defined as:

A set of geometrically identical edges, where each edge forms part of the boundary representation of a different space. See figure 6-5.

A node is defined as:

A set of geometrically identical vertices, where each vertex forms part of the boundary representation of a different space. See figure 6-6.

#### 6.4 Topological constraints

For this building representation, parallels can be drawn with the topological constraints for an individual embedded graph that ensure a boundary representation is consistent (see 5.3). If a space is considered to be a higher dimensional entity, contiguous in 3D, whose boundary of faces, edges and vertices is defined by its constructions, multi-edges and nodes, then the following should be true:

- a) Spaces may only intersect at shared constructions, multi-edges or nodes.
- b) The spaces around each multi-edge form a single circuit.
- c) A path must exist between any spaces of the representation, where the path is formed by crossing shared constructions.
- d) Each construction must belong to exactly 2 adjacent distinct spaces, and be referenced in opposite directions.
- e) Constructions may only intersect at shared multi-edges or nodes.
- f) The spaces around each node form a continuous 'sphere'.

The above is true if the outside of the building model is defined as an external infinite space, whose boundary is the envelope of the building.



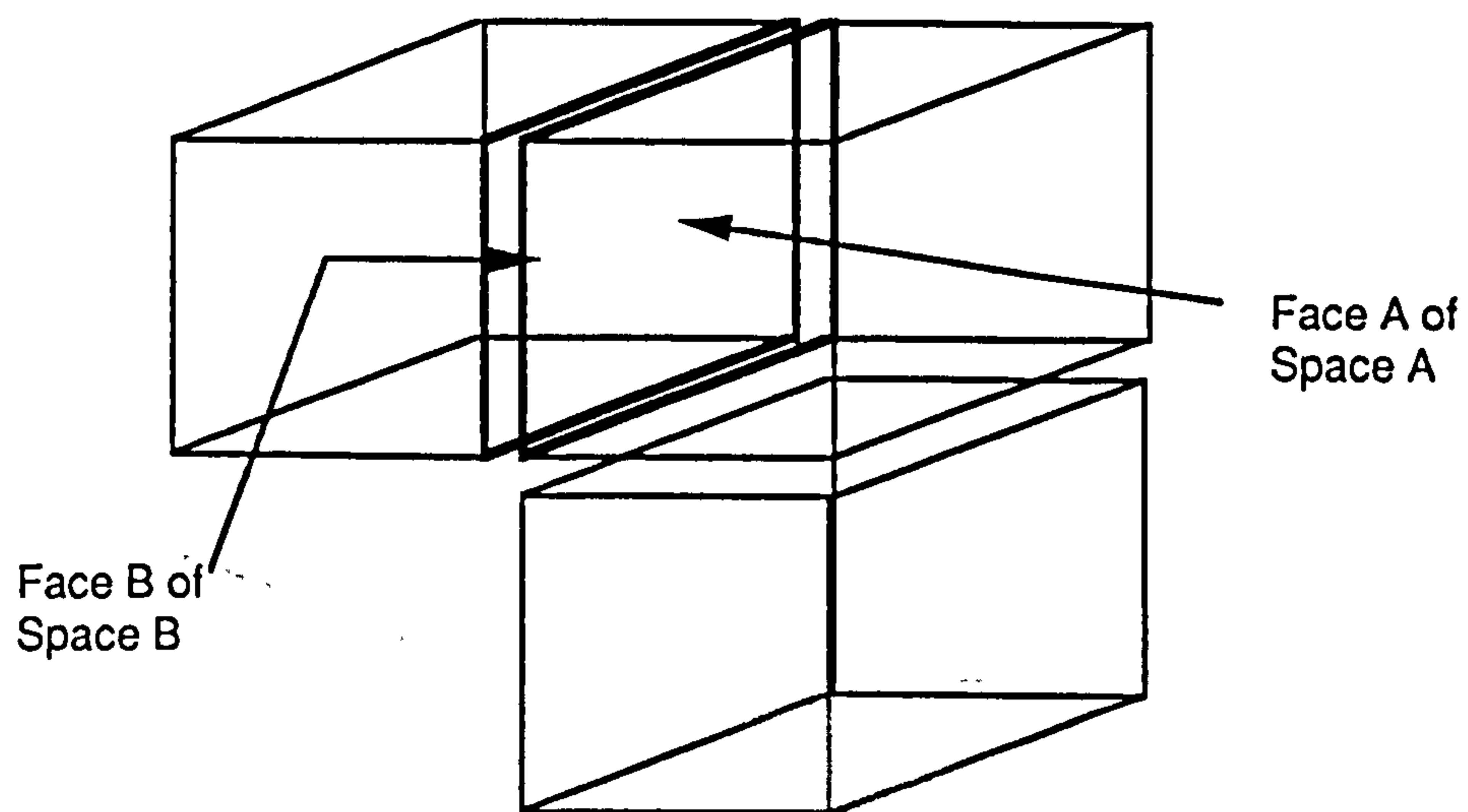


Figure 6-4 Faces of a Construction

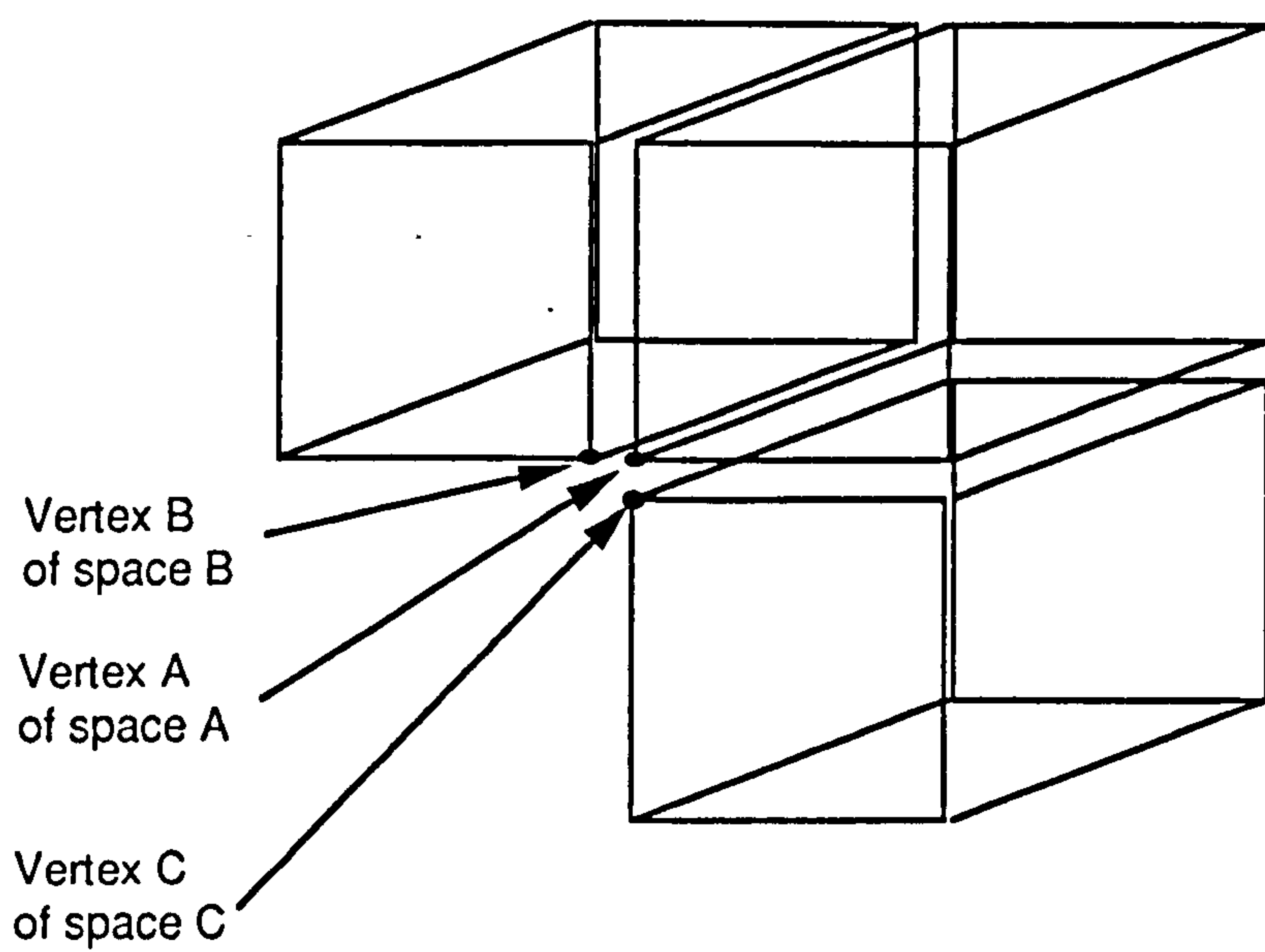
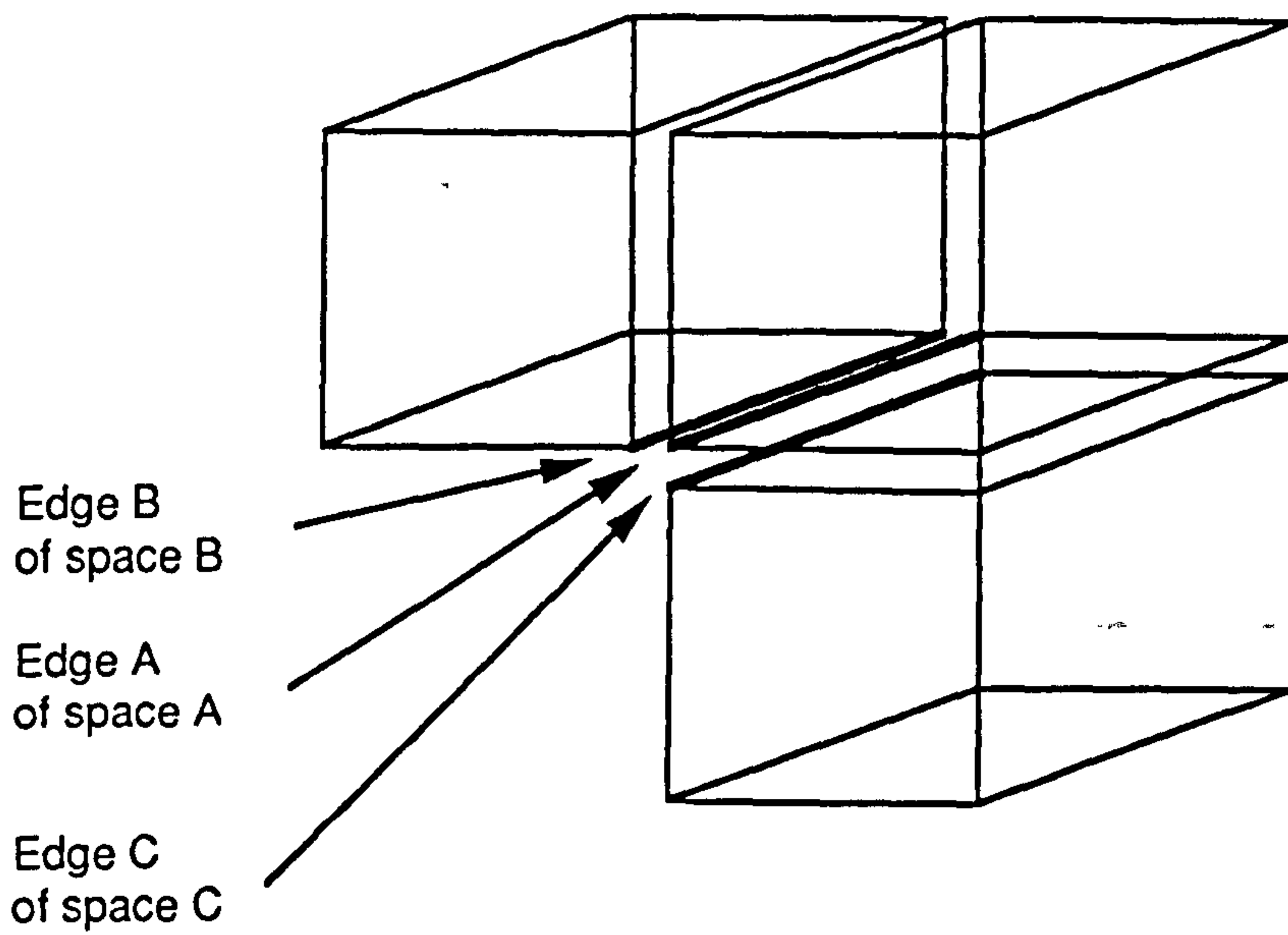


Figure 6-5 Vertices of Node



**Figure 6-6**      **Edges of a multi-edge**

### 6.5 Data Structures and Adjacency Relationships

The winged-edge data structure and other data structures representing an embedded graph have been derived by considering all possible adjacency relationships between the entities: faces, edges and vertices. A data structure is sufficient if all such relationships are explicitly represented or can be derived from it.

Therefore a data structure is required that represents the topological adjacency relationships between spaces, constructions, multi-edges and nodes.

The relationships are summarised in table 6-1.

	Space	Construction	Multi-edge	Node
Space	Spaces adjacent to the space sharing a construction.	Constructions forming the boundary of the space	Multi-edges forming the boundary of the space.	Nodes forming the boundary of the space.
Construction	Two spaces adjacent to the construction.	Adjacent constructions sharing a multi-edge.	Multi-edges forming the boundary of the construction.	Nodes forming the boundary of the construction.
Multi-edge	Spaces with the multi-edge on their boundary.	Constructions with the multi-edge on their boundary.	Multi-edges sharing either of its two end nodes.	Two nodes of the multi-edge.
Node	Spaces with the node on their boundary.	Constructions with the node on their boundary.	Multi-edges connected to the node.	Nodes sharing a multi-edge with the node.

Table 6-1: Adjacency Relationships

### 6.6 Geometry of Building Model

Before attempting to define a data structure for spaces, constructions etc, the type of three-dimensional geometry that needs to be represented in a building model is considered.

Buildings are generally rectilinear in shape. They can usually be modelled with planar surfaces and straight line edges. Any curvature can be approximated by planar facets. Also the thermal analysis assumes all surfaces are planar.

In the same way that geometry is associated with the topological entities, faces, edges and vertices, the geometry needs to be associated with the constructions, multi-edges and nodes, as shown in table 6-2.

Topology	Geometry
Construction	Plane defined by normal direction and co-ordinates of point on plane
Multi-edge	Straight line defined by direction and co-ordinates of point on line
Node	3D co-ordinates of point

Table 6-2: Association of Geometry with Topological Entities

The co-ordinates defining a node are in fact sufficient to specify the geometry of the model. The direction of the multi-edge can be calculated from the points of its two nodes. The normal of the faces of a construction can be calculated from the nodes on its boundary.

### 6.7 Derivation of Data Structure

By considering the relationships above it should be possible to derive a data structure where each entity is represented by a group of data consisting of pointers to other entities (as in the winged-edge data structure described in Chapter 5), such that all required adjacency lists can be produced.

The following points are made from consideration of the above relationships table and with respect to the winged-edge representation of an embedded graph:

a) The boundary of one space can be represented by a winged-edge structure. In this implementation, an 'object' would be a space, and there would be just one shell per object. Therefore it is assumed that all adjacency lists between edges, vertices and faces for one space in the required building model can be produced.

b) Considering the definition of constructions, multi-edges and nodes, as representing the multiplicity of faces, edges and nodes, then it should also be possible to derive from the data structure the following relationships:

- i) Faces of a construction
- ii) Edges of a multi-edge
- iii) Vertices of a node

and also the inverse relationships, to determine the higher level entity group of which each winged-edge entity is a member:

- i) Construction of a face
- ii) Multi-edge of an edge
- iii) Node of a vertex

The relationships are now examined in turn in order to identify the references between entities that need to be present in the data structure.



### 6.7.1 Construction and Space Relationships

Constructions and spaces share faces and therefore they form the 'link' in these relationships. If a face is referenced by both a construction and a space, and the face references back to both the construction and space, then it is shown that the required list of adjacent entities can be found.

a) Space - construction relationship: the list of constructions forming the boundary of the given space is required. The process is as follows (see figure 6-7):

i) All faces of a space are found from the linked list of faces in the winged-edge structure of the space.

ii) Each face in this list references its parent construction. Therefore a list of constructions is created.

b) Construction - Space relationship: the two spaces adjacent to the given construction are required. The process is as follows (see figure 6-8):

i) The left and right faces are referenced by the construction

ii) Both faces reference their parent space.  
Therefore the two adjacent spaces are found.

The above relationships can be derived, therefore, if the following reference pointers exist in the data structure. (A single pointer representing a one-to-one relationship is given as '->', and multiple pointers, representing a one-to-many relationship is given as '->>'):

i) space ->> faces: usually implemented in winged-edge structure as pointer to first face, and pointers from each face to next in list.

ii) face -> space: this corresponds to the optional pointer, sometimes included in a winged-edge representation as a pointer back to the parent object.

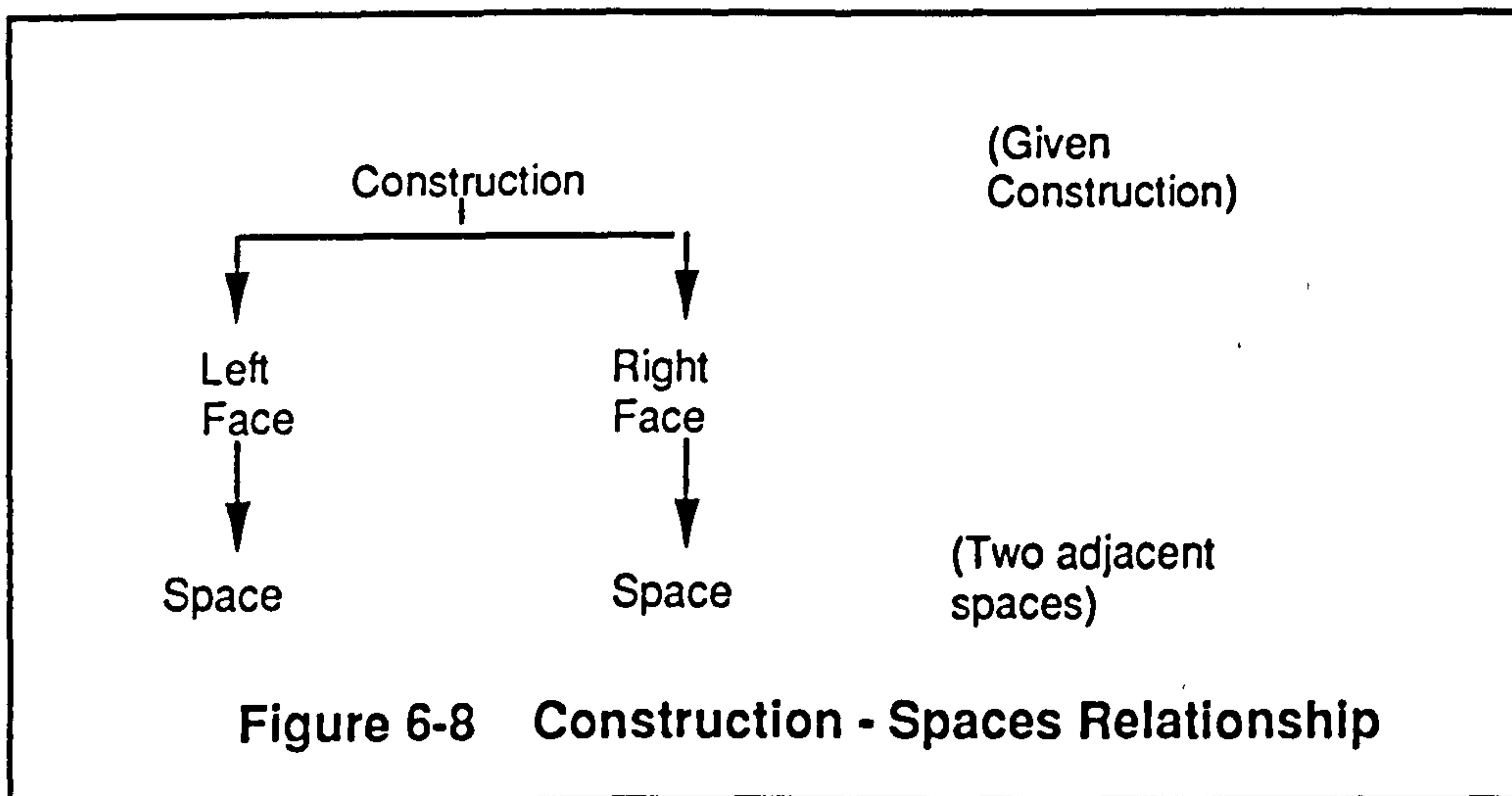
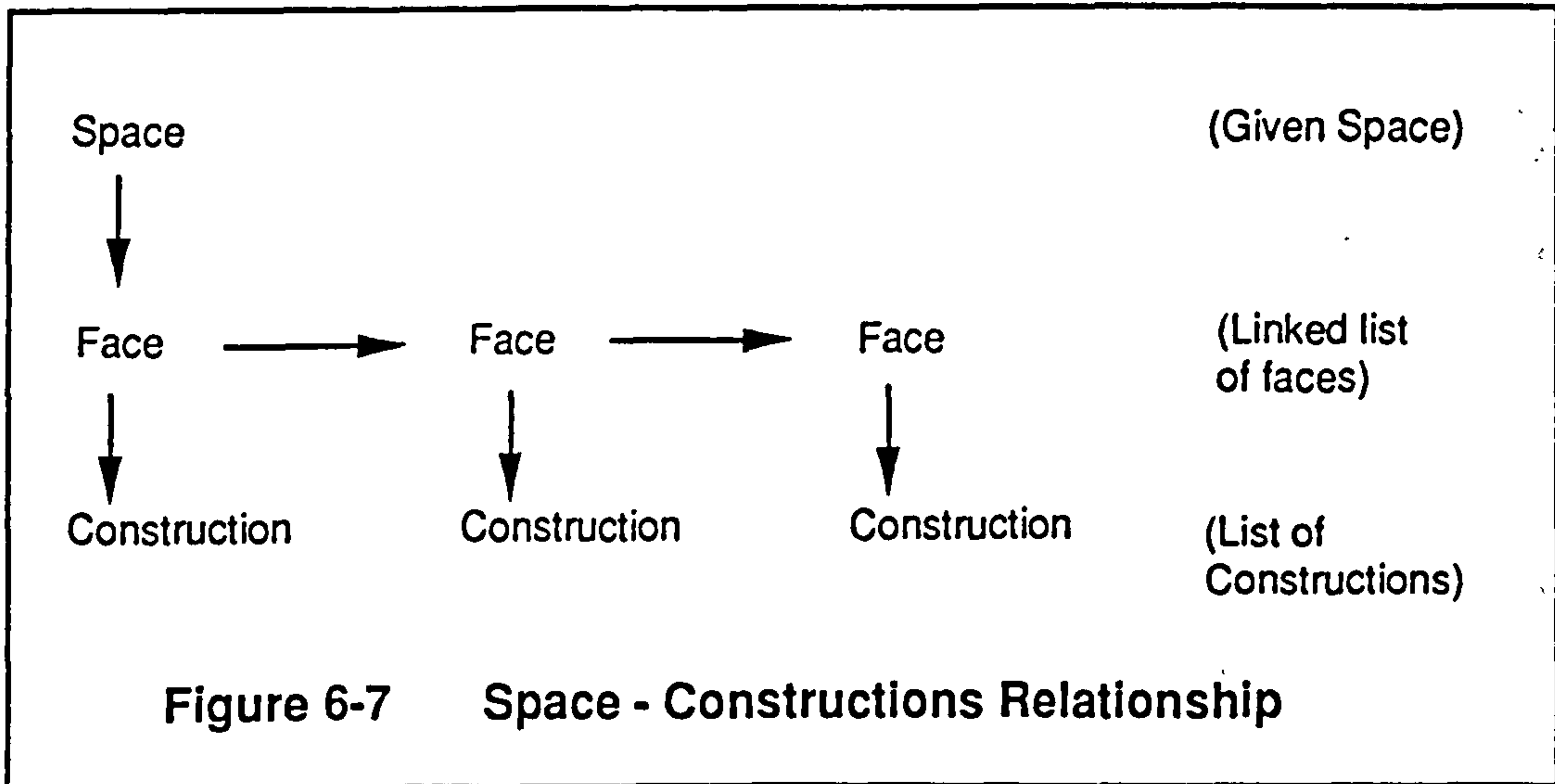
iii) construction ->> left and right faces.

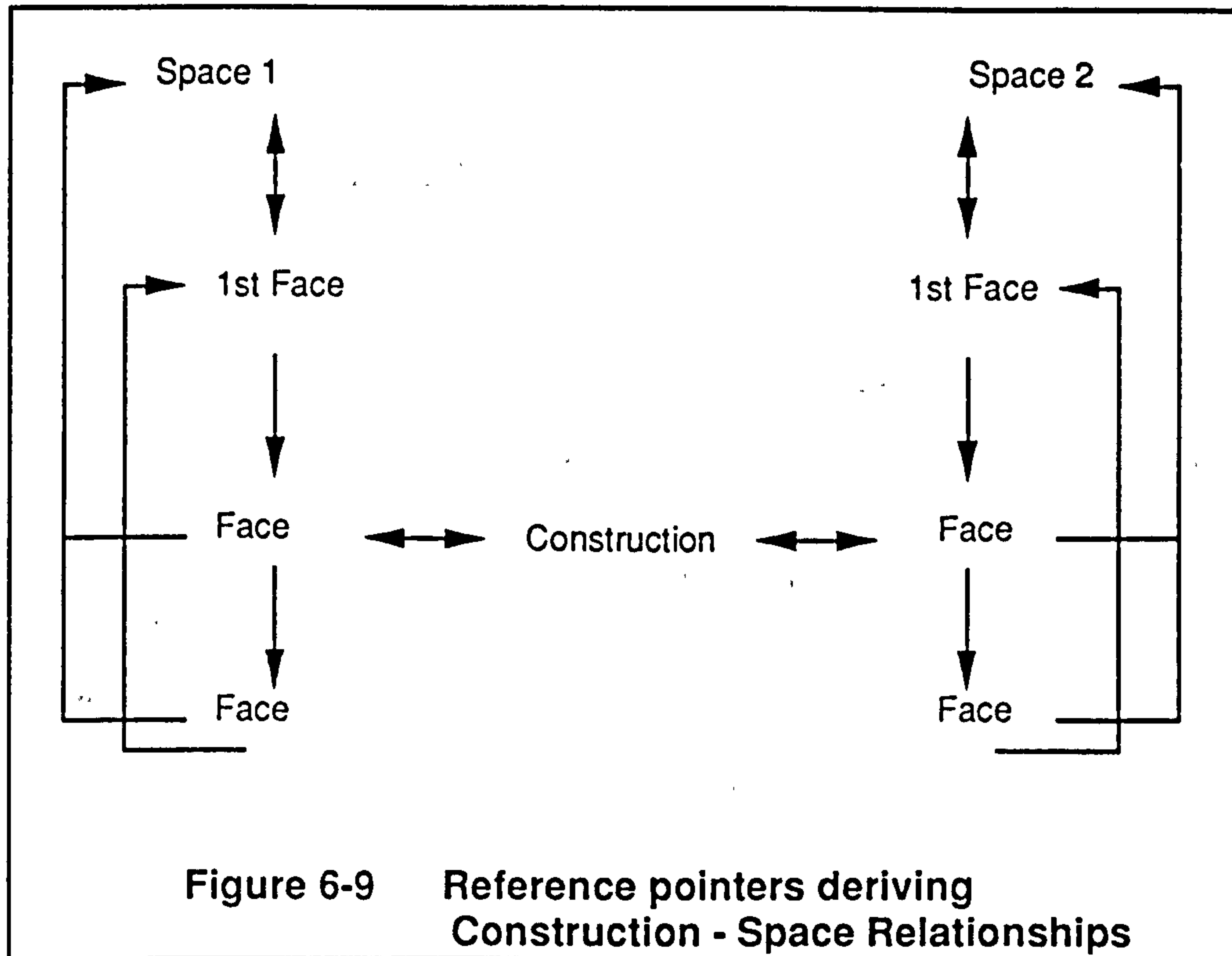
iv) face -> construction.

These reference pointers are illustrated in figure 6-9.

### 6.7.2 Node and Vertex relationships

It is required that the node of a vertex and the vertices of a node can be derived from the data structure. Unlike a construction that always references exactly two faces, a node will need to reference a varying number of vertices. Therefore a 'node' can be defined as a list of 'node instances', where each 'node instance' references a vertex, and the vertex in the winged-edge structure references the 'node instance'.





Therefore a new entity is introduced: a 'node-instance', where the list of these for one node can be implemented by defining a pointer from each instance to the next in the list. The following pointers would be implemented:

- i) node instance -> vertex
- ii) vertex -> node instance
- iii) node instance -> next node instance of list for node (implemented as a circular list, with the last node instance referencing the first).

This is illustrated in figure 6-10.

Note that a node entity itself has not been defined, this is not necessary as long as it is recognised that any node can be identified by one of its node instances.

### 6.7.3 Node - Space Relationship

With the node instance entity implemented as above, the relationships between nodes and spaces can also be found.

- a) Space - Node: the list of all nodes forming the boundary of the space is required. The process is as follows (see figure 6-11).

- i) All vertices of the space can be found from the winged-edge representation of the space.
  - ii) Corresponding node-instance of each vertex is found.

Therefore a list of node-instances is found, each identifying a different node.

- b) Node - Space: The adjacent spaces with the node on their boundary are required. The process is as follows (See figure 6-12).

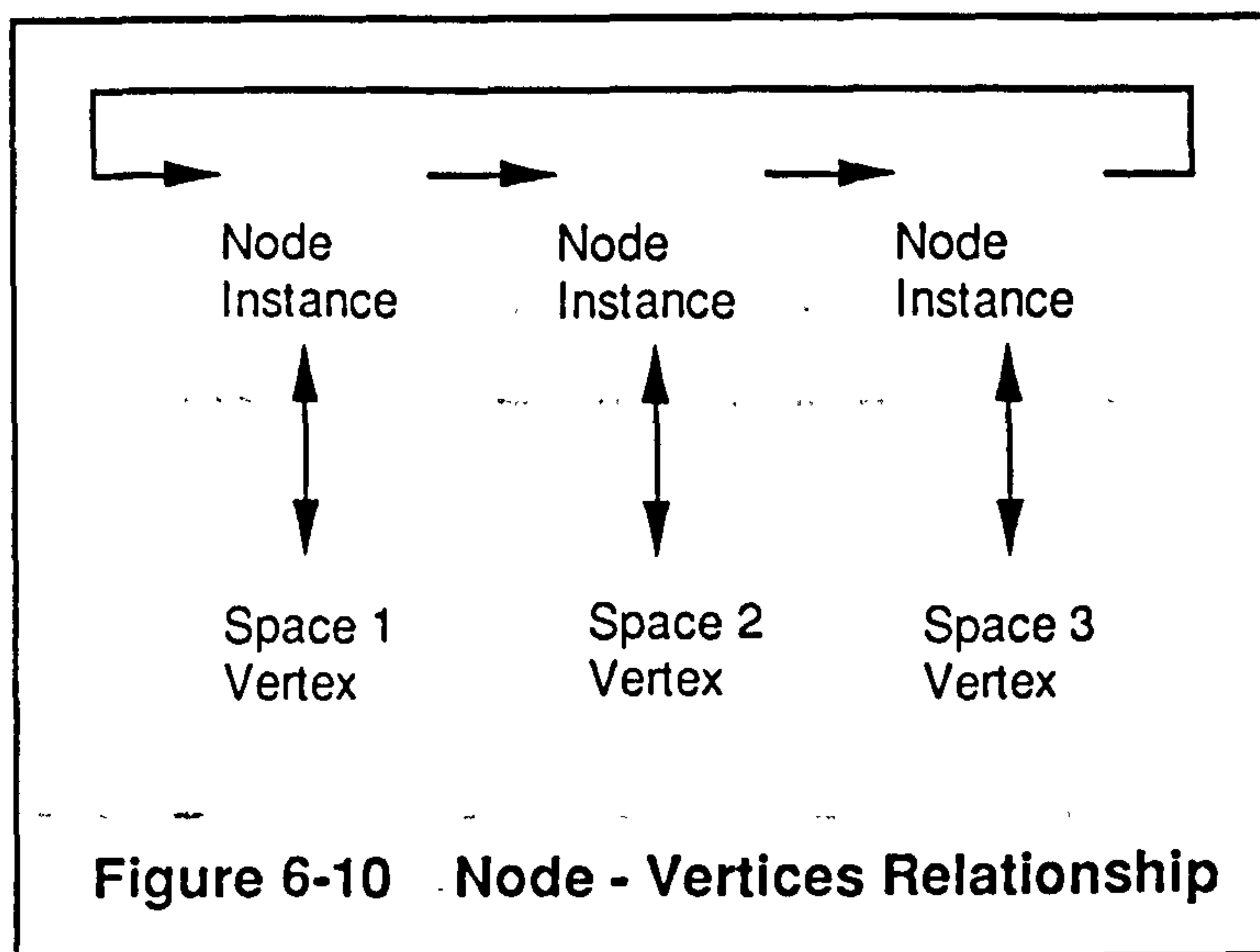
- i) Each node instance of node gives a vertex of the node.
  - ii) The parent space of each vertex is found.

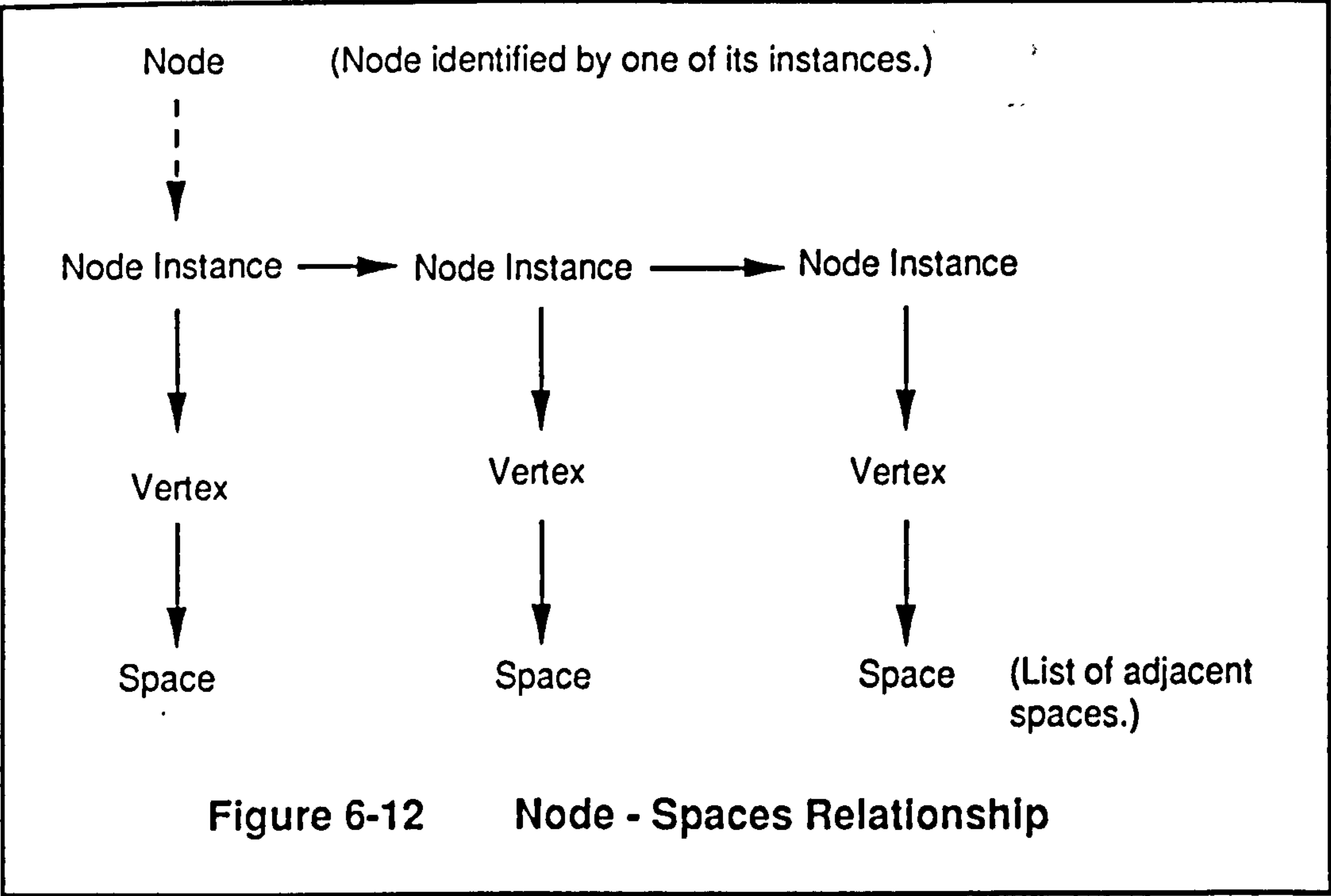
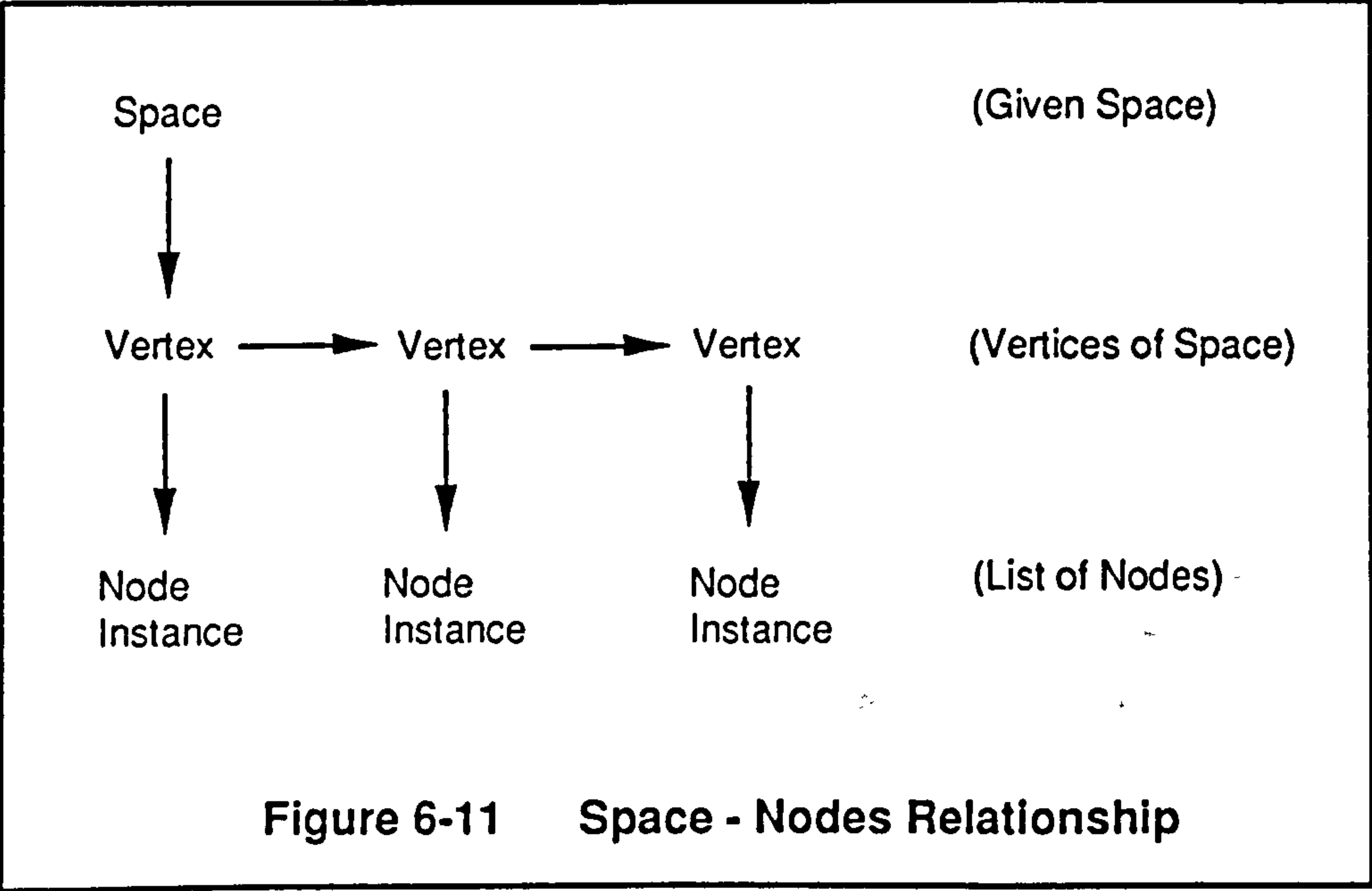
Therefore the list of spaces is found.

To summarise, therefore, the above relationships can be derived if the following pointers exist in the data structure, in addition to the pointers specified in 6.5.2:

- i) space ->> vertices: can be found indirectly from faces of space, or directly by implementing a linked list of vertices of space, pointer from each vertex to the next for the space.
- ii) vertex -> space: optional pointer, sometimes included in a winged-edge representation as a pointer back to the object.







#### 6.7.4 Node - Construction Relationships

Similarly, with the node instance entity implemented as above, the relationships between nodes and constructions can also be found.

a) Construction - Node: all the nodes forming the boundary of the construction are required. The process is as follows (see figure 6-13):

- i) Construction indicates two faces, consider one - either can be used as both would generate the same list of nodes.
- ii) All vertices of face can be found indirectly from winged-edge.
- iii) Corresponding node-instance of each vertex is found.

Therefore list of node-instances is found.

b) Node - Construction: The list of adjacent constructions with the node on their boundary is required. The process is as follows (see figure 6-14).

- i) Each node instance of node gives a vertex of the node.
- ii) The list of faces adjacent to the vertex can be found indirectly from the winged-edge.
- iii) The parent construction of each face is found.

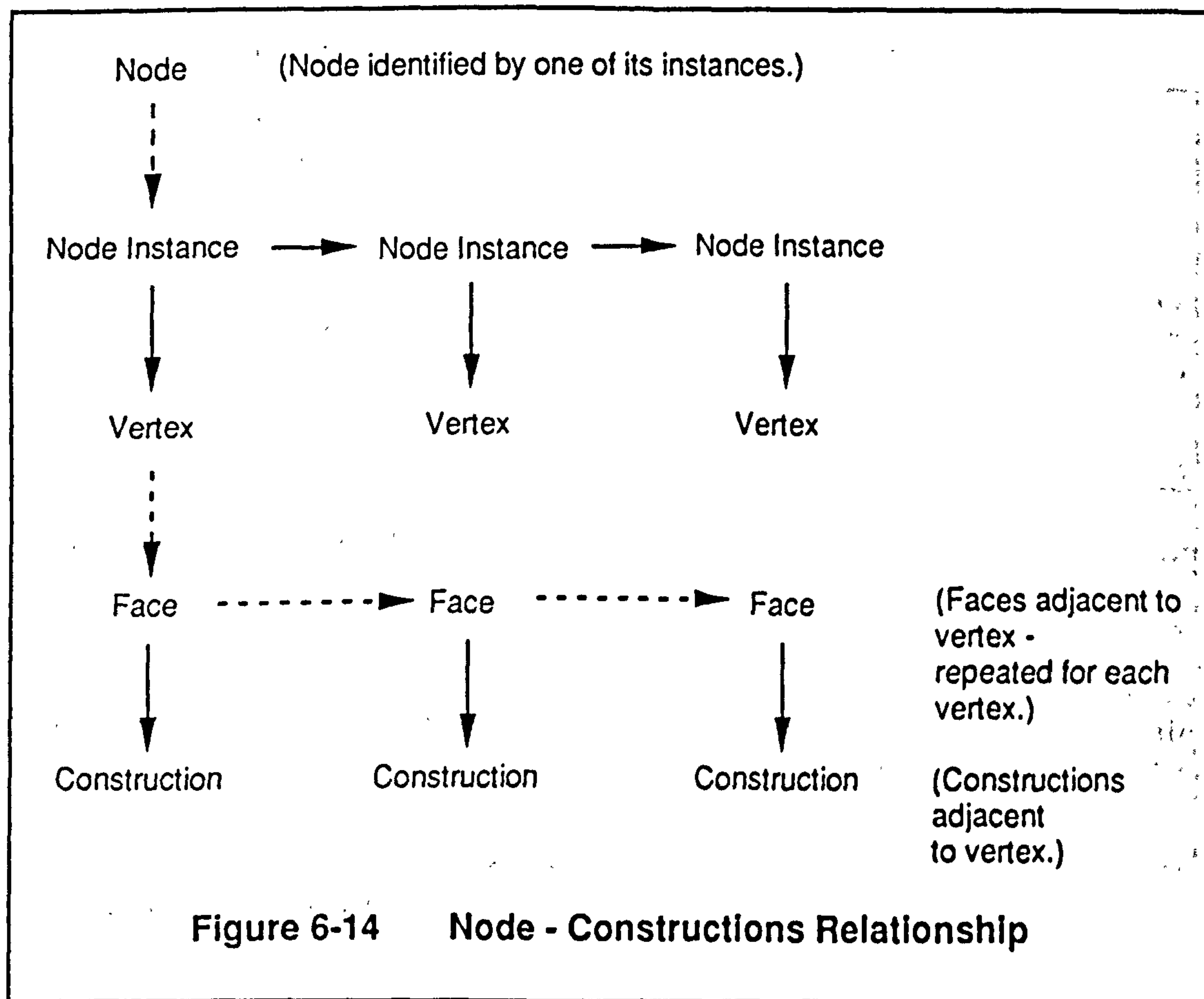
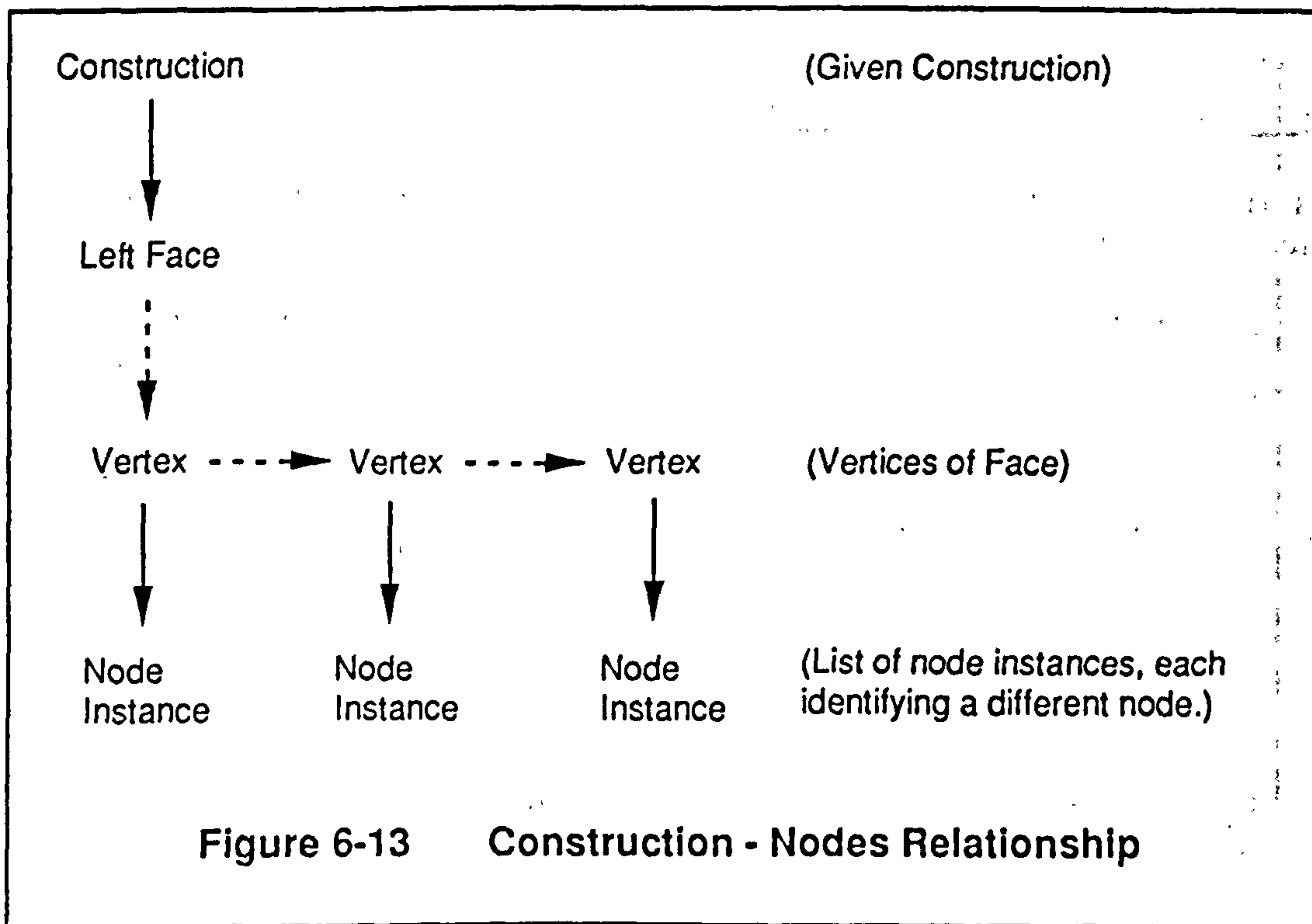
Therefore a list of constructions is found. This method will result in duplication of constructions in the list, where the left face of the construction is referenced from one space, and the right from another space.

To summarise, therefore, with the pointers specified in 6.5.1 and 6.5.2, the above relationships can be derived with no further pointers in the data structure.

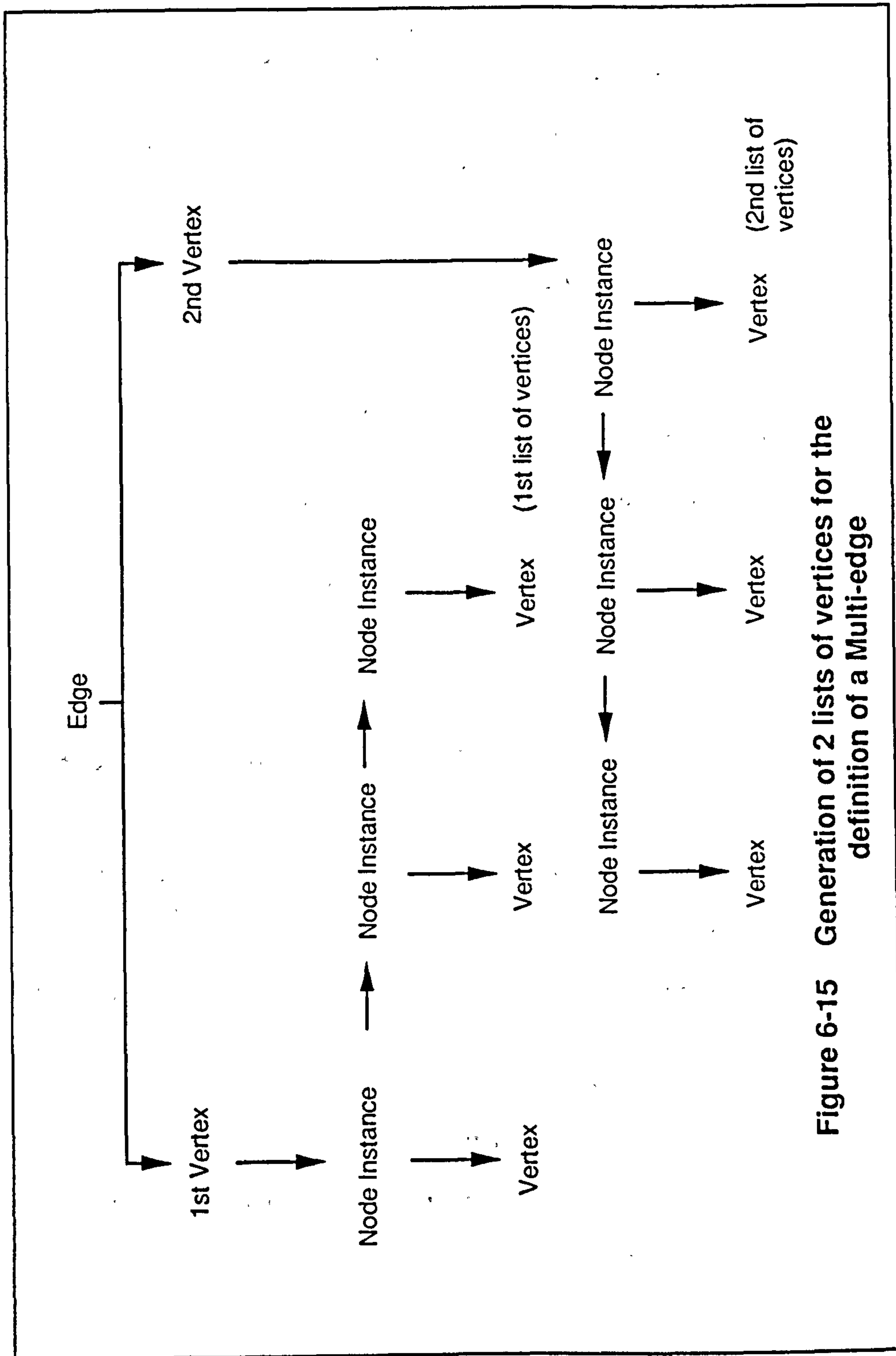
#### 6.7.5 Multi-edge - Edge Relationships

A multi-edge is defined as the group of edges which connect two nodes, where each edge belongs to the winged-edge structure of a different space. As stated previously, the geometry of the edges are all straight lines. It is therefore possible to derive the edges of a multi-edge from the references to nodes and vertices, by comparison. That is, given one edge, all other edges forming the multi-edge entity can be found by the following process (illustrated in figure 6-15):

- i) Get both vertices of edge.
- ii) Get node instances of both vertices
- iii) For each node instance, get list of node instances of node.
- iv) Get vertex of each entry in both lists of node instances.







**Figure 6-15** Generation of 2 lists of vertices for the definition of a Multi-edge

This generates two lists of vertices. By finding edges which connect a vertex in one list with a vertex in the other, all the required edges of the multi-edge are found:

Consider each vertex from first list:

- i) Get all edges of vertex: indirectly from winged-edge.
- ii) For each edge, find opposite vertex.
- iii) Search for this vertex in 2nd list of vertices
- iv) If vertex is found then edge is a required edge i.e. a multi-edge instance of the multi-edge.

Therefore, the assumption is made that a list of multi-edge instances defining a multi-edge can be derived from any edge i.e. a multi-edge can be identified by any of its edges.

#### 6.7.6 Space - Multi-edge Relationships

These relationships can be derived as follows.

a) space - multi-edge: the list of multi-edges forming the boundary of the space is required. The process is as follows:

- i) All edges of a space are found from the winged-edge representation of the space.
- ii) Therefore a multi-edge is implicitly identified by each edge.

b) multi-edge - space: all spaces with the multi-edge on their boundary is required. The process is as follows (see figure 6-16):

Again assuming that a multi-edge is identified by one of its edges:

- i) The list of edges forming the multi-edge is found.
- ii) Parent space of each edge in instance list is identified.

Therefore the required list of spaces is found.

To summarise, the above relationships can be derived if the following pointers exist in the data structure:

- i) space ->> edges: can be found indirectly from faces of space, or directly by implementing a linked list of edges of space, pointer from each edge to the next for the space.
- ii) edge -> space: optional pointer, sometimes included in winged-edge as pointer back to object.

### 6.7.7 Multi-edge - Construction Relationships

Similarly, the multi-edge and construction relationships can be derived assuming the generation of a multi-edge instance list.

a) Construction - Multi-edge: all multi-edges forming the boundary of the construction are required. The process is as follows (see figure 6-17):

- i) Construction indicates two faces, consider one - either can be used as both would generate the same list of multi-edges.
- ii) All edges of face can be found indirectly from winged-edge.
- iii) Therefore a multi-edge is identified by each edge.

b) Multi-edge - Construction: The list of adjacent constructions with multi-edge on their boundary is required. The process is as follows (see figure 6-18):

Assuming that a multi-edge is identified by one of its edges:

- i) Get multi-edge instance list
- ii) The left and right faces adjacent to each edge can be found indirectly from the winged-edge.
- iii) The parent construction of each face is found.

Therefore a list of constructions is found. This method will result in duplication of constructions in the list, where the left face of the construction is referenced from one edge, and the right from another edge.

Therefore, with the pointers specified in 6.5.1, the multi-edge and construction relationships can be derived with no further pointers in the data structure.

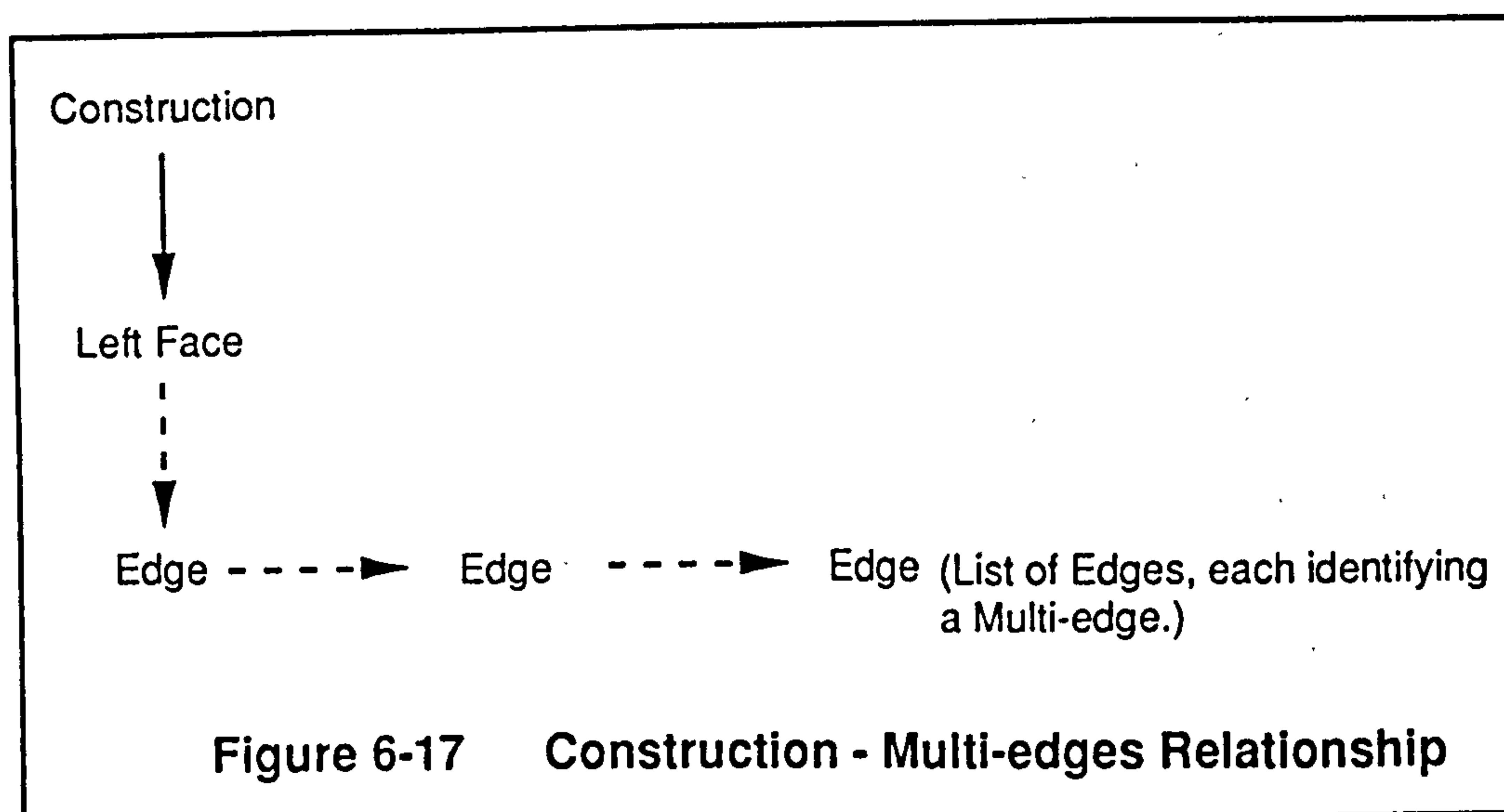
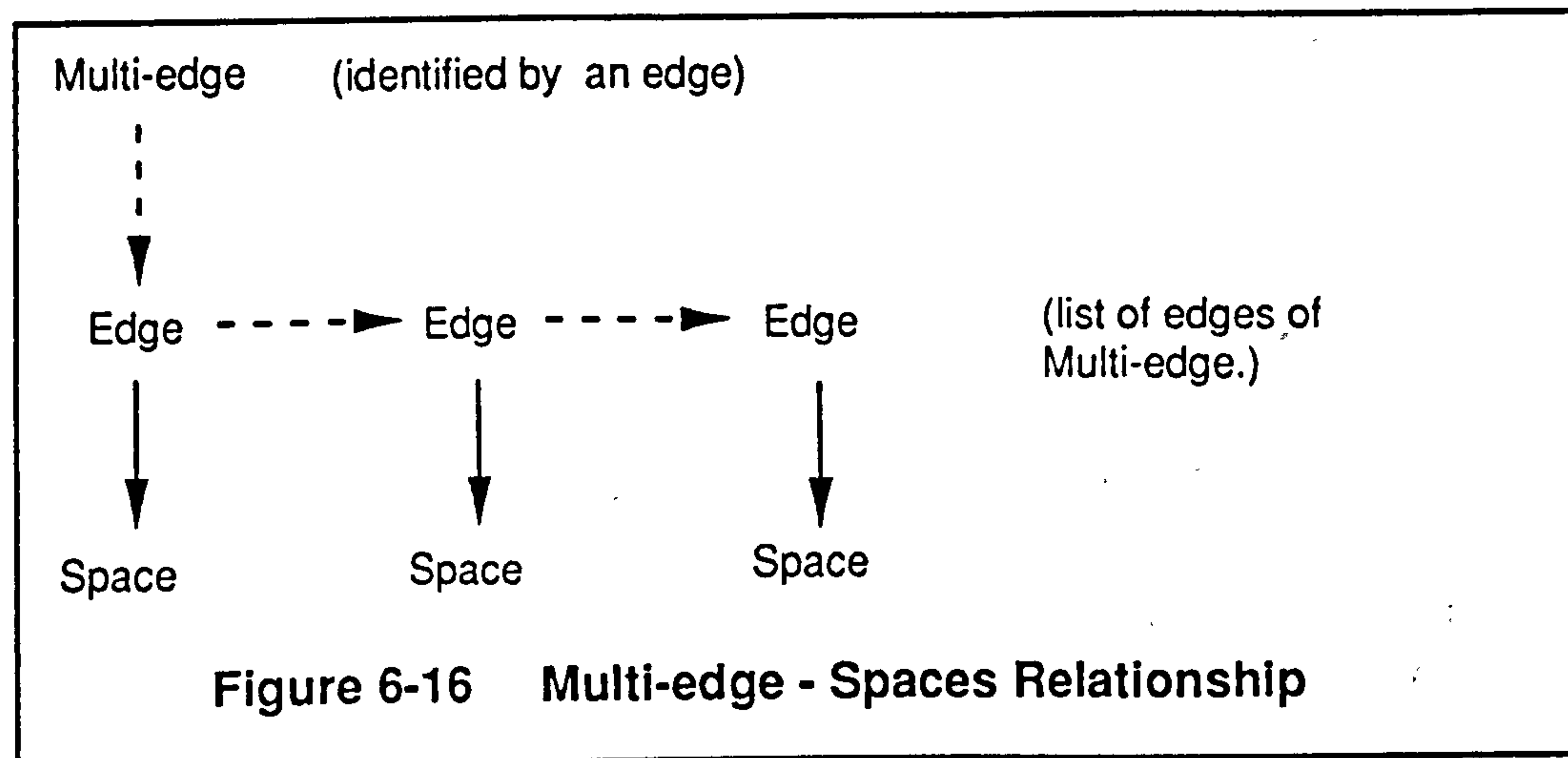
### 6.7.8 Multi-edge - Node Relationships

These relationships are dependent on the fact that a node is identified by one of its node instances, and a multi-edge is identified by one of its edges.

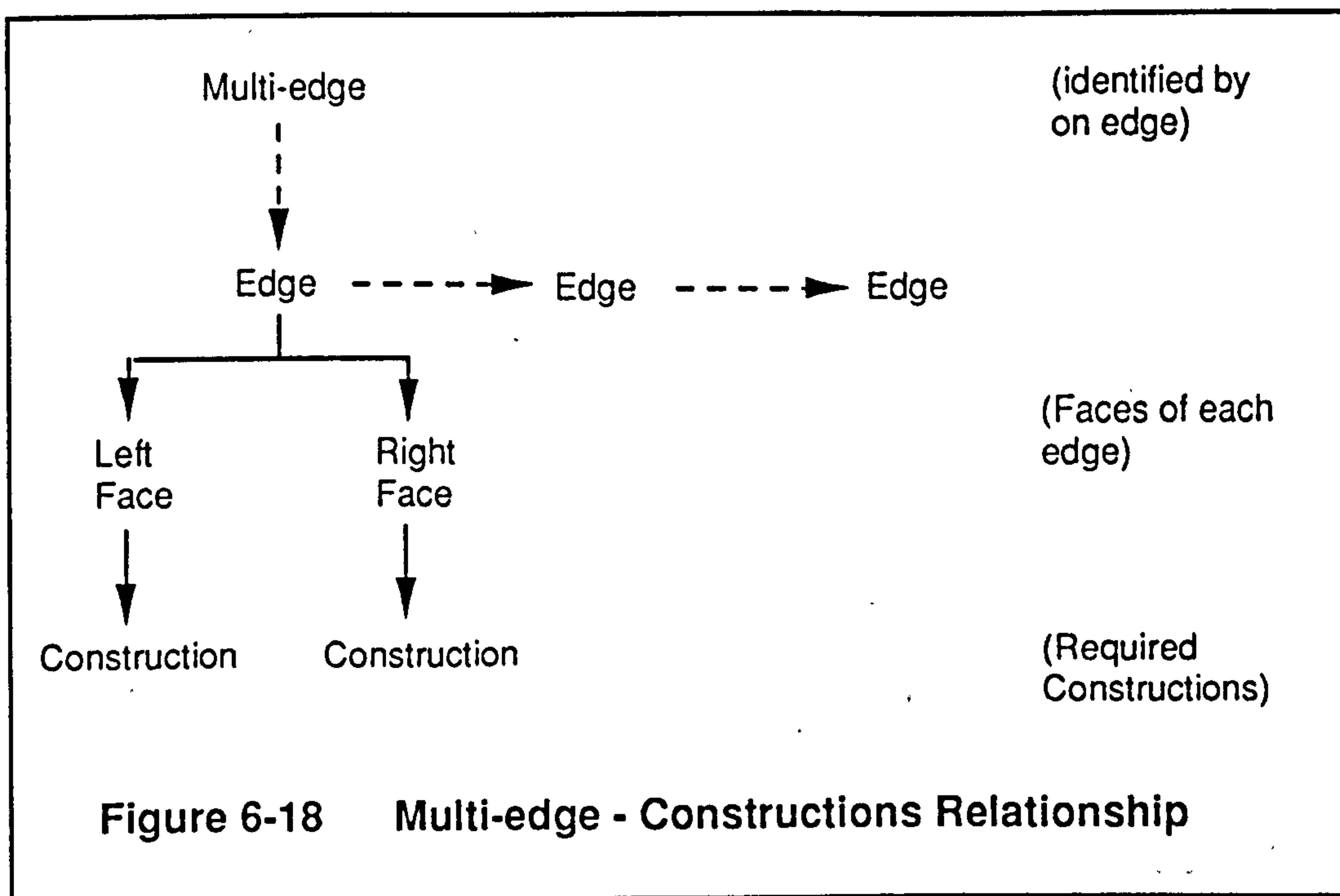
a) Multi-edge - node: the two nodes of a multi-edge are required. The process is as follows:

- i) Get two vertices of edge.
- ii) Node-instance of both vertices identifies the two nodes.

N.B. it is not necessary to generate the multi-edge instance list.







b) Node - multi-edge: the list of multi-edges connected to the node is required. The process is as follows:

- i) Get list of node instances of node
- ii) Get vertex of each node instance
- iii) Get edges of each vertex - indirectly from winged-edge data.

The lists of edges identify the multi-edges. This will produce duplication of multi-edges. Therefore assuming that the relationship 'edge a belongs to the same multi-edge as edge b' can be determined true or false, then the duplication can be removed.

For 'edge a belongs to same multi-edge as edge b' to be true, then where a1 and a2 are the two vertices of edge 'a', and b1 and b2 are the two vertices of edge b, the following must be true:

either

'a1 and b1 belong to the same node' and  
'a2 and b2 belong to the same node'

or

'a1 and b2 belong to the same node' and  
'a2 and b1 belong to the same node'

Therefore a further relationship is defined, which must be determined true or false: 'node-instance of vertex a belongs to the same node as node-instance of vertex b'. This can easily be determined by searching through the linked list of node instances starting from 'node instance of a', if 'node instance of b' is found then relationship is true.

### 6.7.9 Self relationships

The remaining relationships can be seen to be derived from combinations of the above.

a) Space - Space relationship: the list of spaces adjacent to the given space, sharing a construction is required and is derived from the following relationships:

- i) Space - construction relationship provides list of constructions.
  - ii) Construction - space relationship provides two adjacent spaces of each construction.
- Therefore, the other space to the given space in each pair of spaces provides the list of spaces.

b) Construction - construction: the list of adjacent constructions sharing a multi-edge is required and is derived from the following relationships:

- i) Construction - multi-edge derives list of multi-edges
- ii) Multi-edge - construction derives list of constructions of each multi-edge. Original construction can be ignored in each list.

c) Multi-edge - multi-edge: the list of multi-edges sharing either of the given two end nodes of the multi-edge is required and is derived from the following relationships:

- i) Multi-edge - node derives two nodes of multi-edge.
- ii) node - multi-edge derives multi-edges of each node, identified by an edge of each multi-edge. Original multi-edge can be ignored in list, assuming relationship described above: 'edge a belongs to the same multi-edge as edge b' can be determined true or false.

d) Node - node: the list of nodes sharing a multi-edge with the given node is required and is derived from the following relationships:

- i) node - multi-edge derives multi-edges of each node, identified by an edge of each multi-edge.
- ii) multi-edge - node derives two nodes of each multi-edge, identified by a node-instance of each node. Required nodes are those opposite to given node.

## 6.8 Summary of Data Entities

From the above discussion, it is now possible to identify the references that each data entity must make to other data entities, and define a data structure of records and fields.

As indicated, there is no need to define a multi-edge entity explicitly, all the required relationships can be derived. Although some of these derivations may be inefficient, in that duplication of entities arise, those relationships involving multi-edges are not frequently required.

In addition there is the overhead of maintaining additional data entities, when creating and modifying the model, ensuring that all data and references are consistent.

The building is the highest level entity in the hierarchy, and is represented by a set of spaces, and a set of constructions.

### 6.8.1 Representation of Geometry

As stated previously the geometry of the model can be sufficiently defined by the 3D points: a node must refer to one 3D point. As there is no single node

entity, each node instance of a node refers to the same point entity. A point entity is defined which consists of the x,y, and z co-ordinates of a point.

Although it is not necessary, it is advantageous for each construction entity to refer to the normal of one of its faces, the normal of the other face being in the opposite direction, and easily derived. The normal of a face is required frequently: in particular it is used to derive the orientation and slope of zone surfaces.

### 6.8.2 Definition of Data Records

The data record for each entity is described in tables 6-3 to 6-11, with possible field names for each data item.

The entities in tables 6-8 to 6-11 have all been previously defined in the winged-edge structure. The new fields, indicated with \*, are either referring to the new entities above, or are additional pointers to aid in accessing the winged-edge data. The Object entity has been replaced by the Space entity and the Shell entity is not necessary as a Space can only have one boundary.

#### **BUILDING**

Field Name	Description
space	1st space of building

Table 6-3: Building Data Record

#### **SPACE**

Field Name	Description
nextspace	Next space in list for building
face	1st face of space

Table 6-4: Space Data Record



**CONSTRUCTION**

Field Name	Description
nextcon	Next construction in list for building
lface	'Left' face of construction.
rface	'Right' face of construction.
normal	3 components of normal direction of left face.

Table 6-5: Construction Data Record**NODE INSTANCE**

Field Name	Description
point	3D point of node
vertex	Corresponding vertex of winged-edge structure.
nextnode	Next node instance in list for node

Table 6-6: Node Instance Data Record**POINT**

Field Name	Description
x	x ordinate of point
y	y ordinate of point
z	z ordinate of point

Table 6-7: Point Data Record

**EDGE**

Field Name	Description
pvert	1st vertex of edge - 'Previous vertex'
nvert	2nd vertex of edge - 'Next vertex'
lloop	Loop of left face adjacent to edge
rloop	Loop of right face adjacent to edge
elcc	Edge counter-clockwise from edge in left loop
elcw	Edge clockwise from edge in left loop
ercc	Edge counter-clockwise from edge in right loop
ercw	Edge clockwise from edge in right loop
nextedge *	Next edge of winged-edge structure

Table 6-8: Edge Data Record**FACE**

Field Name	Description
space *	Parent space of face
construction *	Construction of face
nextface	Next face in list for shell.- Circular linked list
loop	1st loop of face - usually peripheral loop

Table 6-9: Face Data Record

LOOP

Field Name	Description
nextloop	Next loop in list for face. Circular linked list.
edge	An edge of the loop, or NULL pointer if there is no edge.
vertex	Single vertex of loop, if loop has no edges.

Table 6-10: Loop Data Record

VERTEX

Field Name	Description
edge	An edge attached to the vertex, or NULL pointer if there is no edge.
loop	Loop of single vertex, if vertex has no attached edges.
nextvert *	Next vertex of winged-edge structure.
node *	Node instance of vertex.

Table 6-11: Vertex Data Record





## 7. CREATION OF BUILDING DATA STRUCTURE

As discussed in Chapter 5, the components of a modelling system are:

- a) Modelling Operations: Input
- b) Model: Data Structure
- c) Access of Model: Output

In Chapter 6, a basic data structure was described for modelling a building. It was ensured that all relations between entities could be derived. The assumption can be made therefore that all necessary functionality to access the data structure could be provided. More important however is the set of modelling operations required for the input of the model, to create the data structure and ensure its consistency.

There are several levels of functionality involved in creating a modelling data structure:

- a) The user interface is the highest level, where the data structure is completely hidden from the user. The model is created by modelling methods which reflect the user's expertise and the way in which the problem is presented to the user.
- b) The actions from the user interface are then interpreted into high level operations. For example, the definition of primitive shapes, boolean operations and profile sweep are operations available in solid modelling systems. Normally, in a solid modelling system, these operations ensure the consistency of the geometry. As discussed previously, Euler operators generated by these high-level operators are purely topological operators, where geometry is specified separately and attached to the topology.
- c) The low level operations, such as Euler operations, act directly on the underlying data structure, to create the faces, edges, loops and vertices of a winged-edge structure. These operations ensure the consistency of the topological data.

These levels of functionality can be considered with regard to creating the building model:

- a) The user in this application of environmental design is likely to be a Building Services Engineer, who will be given details of the design of a building. The user interface needs to provide the functionality to define the geometric form of a building, most probably in terms of walls, floors, ceilings, roofs and windows, positioned in 3D space. As discussed in Chapter 4, Building Modelling systems do not usually identify a room or a space entity explicitly. However, it is necessary here, in particular the user will need to be able to identify a displayed space entity, to allow attributes to be assigned to it, i.e. a zone number.
- b) The next level of operation must take this geometric data and interpret it into the required topological entities. Low level operations are then generated to create constructions, nodes

and spaces. The geometry of the constructions, nodes and spaces would need to be specified here and its consistency ensured.

c) The lowest level of operation should provide the functionality to create the constructions, nodes and spaces, ensuring their topological consistency, and attach the specified geometry. The operation to create a single space could be similar to any of those suggested in ii) above for a solid modelling system.

At a lower level again, therefore, the standard Euler operations can be used to create the topology of an individual space.

## 7.1 Building Geometry

It is necessary to consider the geometric features of a 3D Building Model. It has so far been stated that planar facets are adequate to model a building. In addition, buildings are generally  $2\frac{1}{2}$  dimensional in form: each floor consists of vertical walls placed on the same floor level, of the same height, with spaces that have horizontal floors and ceilings. CAD systems designed primarily for modelling buildings (see Chapter 4) usually organise their data of instanced components on a floor by floor basis. The main exception to this are roof shapes which tend to have sloping planar facets.

A major requirement in building modelling, therefore, is in positioning walls onto a horizontal plane specified by a height, which is the floor level. A wall can be modelled by two endpoints in plan i.e. 2-dimensional, a height and a thickness.

Sweep operations are usually used in solid modelling to create the boundary representation of  $2\frac{1}{2}$ D objects: a 2-dimensional profile, is swept by a specified distance, usually in a direction normal to the plane of the profile.

As indicated above, at the lowest level of operation, a solid modelling method could be used to create a single space. Here, a sweep operation could be used to generate its boundary where a profile was determined from the 2-dimensional representation of its surrounding walls. A sweep operation can be specified in terms of Euler Operations. This would ensure the consistency of the winged-edge data representation for an individual space, and ensure that enclosed spaces are created.

However, in parallel to creating spaces, the construction and node entities also need to be created. This is addressed if the sweep-type operation is specified sufficiently at the higher level of operation in terms of a 2D representation, which is not just a single profile, and a sweep direction and distance, such that the required constructions, nodes and spaces can be determined.

## 7.2 Planar Graphs

It is important to recognise here, the fact that a 2-dimensional floor plan of a building can be represented by a planar graph. A planar graph is a graph which can be drawn on a plane such that no two of its edges intersect. A graph which can be embedded onto the surface of a sphere is also a planar



graph, that is a graph with a genus of zero. The edges of the graph represent the walls, and the faces represent the spaces. See figure 7-1.

This fact has been mainly used in the problem of space layout for a floor by considering the dual of this graph, where the vertices represent the spaces, when solving for a particular space adjacency requirement (21,22).

It can also be seen that any section, horizontal or vertical, through the desired building model is actually a 2-dimensional planar graph, where the edges represent a cross-section through a construction, the faces are the cross-section through the spaces and the vertices are the cross-section through a multi-edge.

The topology of a planar graph can be created by Euler Operations. By considering this representation of a construction by an edge, a space by a face, and a multi-edge by a vertex, it can be seen that there is an analogy between Euler operations in terms of edges, faces, loops and vertices, and similar operations in terms of constructions, spaces and multi-edges.

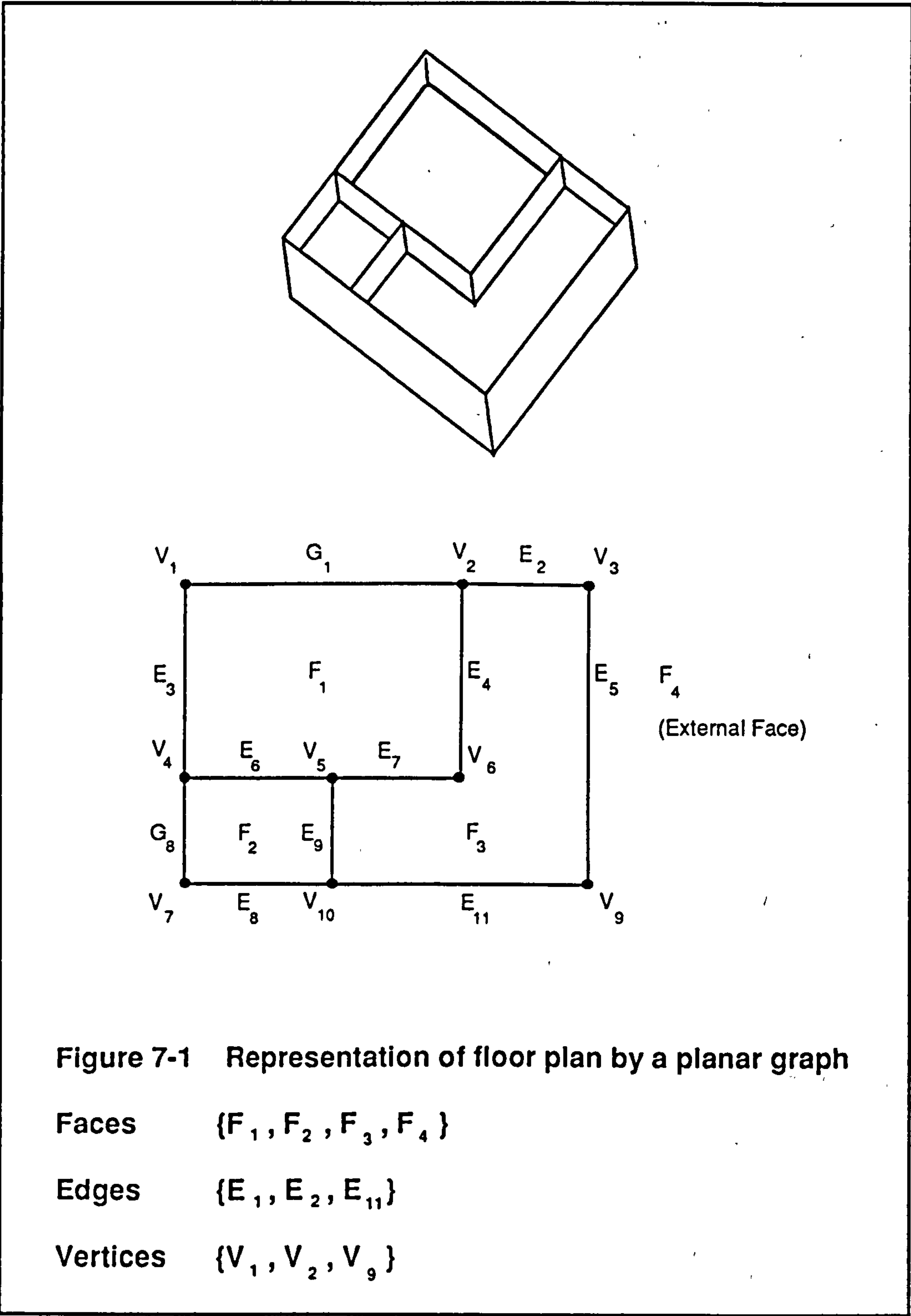
Considering the Euler operation 'make an edge and face' where the edge is connecting two existing vertices, so dividing a face into two, a similar operation can be conceived for a construction: 'make a construction and space', where the boundary of the construction connects a loop of multi-edges on the boundary of the space, so dividing the space into two spaces.

The similar operation 'make an edge and kill a hole loop' is analogous to 'make a construction, kill the genus of a space'. Again the boundary of the construction connects a loop of multi-edges on the boundary of the space. In this case the resulting construction can be compared to a wire edge, that has the same face on either side: the construction has the same space on either side.

To create a multi-edge, the operation 'split construction' is required, equivalent to 'split-edge'. The result is to divide the construction into two, therefore a connected set of multi-edges could be created by describing a curve on the construction.

The equivalent to 'make an edge and make a vertex' would be an operation when a construction was added where not all of its boundary was attaching to existing multi-edges, i.e. 'make a construction, make multi-edge(s)'. Again the resulting construction can be compared to a wire-edge.

The specification of such operations would be quite complex in the general 3-dimensional case, and it can be seen that the 2-dimensional equivalent operations, i.e. Euler operations are sufficient to create a floor planar graph, from which the adjacencies of the represented spaces and constructions can be derived in order to create the actual space and constructions entities.





### 7.3 Modelling Method

From the above discussion, it is seen that a modelling method can be determined which will create a  $2\frac{1}{2}$  dimensional representation of one floor of a building, based on sweeping a 2D representation, such that the required construction, spaces and nodes can be created. In summary the method is as follows:

- i) Position edges defined by 2 endpoints in 2D, to represent the wall constructions of a floor.
- ii) From this geometrical data, determine and generate the Euler operations to create a planar graph, represented by a winged-edge data structure, such that the edges correspond to constructions, faces to spaces, and vertices to vertical multi-edges.
- iii) When the floor layout is complete, sweep the planar graph, using the adjacency data implied in the winged-edge structure representing the graph to create 3-dimensional spaces and constructions.

The creation of the building model is therefore divided into two steps:

- a) Generation of the Euler Operations to create a 2-dimensional representation of a floor.
- b) A single sweep-like operation to create the full building model data structure.

### 7.4 Creation of Planar Graph representing Floor

The Euler operations that are required to create a planar graph representation are now considered. The Euler-type operations described in 7.2 in terms of constructions, spaces and multi-edges, define the required Euler operations. The edge representing a wall construction, is used to create the topology of the planar graph. Therefore all operations for the creation of an edge are necessary:

- i) Make an edge and make a vertex
- ii) Make an edge and make a face
- iii) Make an edge and kill a hole loop

To create the vertices to which the edges can be attached, the following operations are required:

- i) Make a body, vertex and face - to initially create the graph
- ii) Make a vertex and a hole loop
- iii) Split-edge

As there is only one 'body', and the genus of a planar graph is always zero, no other Euler operations are necessary. The Euler equation to be satisfied is simplified to :

$$V - E + F - H = 2$$

The planar graph can be represented by a standard winged-edge data structure as defined in Chapter 5. There is no need to represent multiple shells, and there is only one 'object'.

A face may have more than one loop i.e. a peripheral loop and hole loops. This is feasible in a floor layout, for example, a central block of stairs or lifts may be enclosed by a corridor.

The geometry of the graph is 2-dimensional, therefore no face surfaces need to be specified. As all the edges are straight lines, the only geometry which needs to be included in the data is the 2-dimensional co-ordinates of each vertex.

It is noted that this winged edge representation has slightly different record definitions to those detailed in Chapter 6.

### 7.5 Creation of Building Model from Planar Graph

The process by which the spaces, nodes and constructions of a model can be created from a planar graph representation of a floor by a sweep-like operation is now considered.

The sweep operation has to perform the following:

a) Each face of the graph corresponds to a space. A standard sweep operation will generate the complete winged-edge structure representing a space. This is achieved by creating a profile from the edges of each space, and sweeping the profile by the required floor to floor height.

b) Each graph vertex must generate a vertical multi-edge, i.e. two nodes, a lower node at the level of the original profile and an upper node, at the level of the swept profile. Each node consists of a list of node-instances, where each node-instance corresponds to a graph face adjacent to the graph vertex. The node-instance must reference a vertex of the space created by sweeping the adjacent graph face.

c) Each graph edge must generate a construction. The construction must reference two faces, where these are faces of the two spaces created by sweeping the two graph faces adjacent to the edge.

## 7.6 Result of Modelling Operation

This operation will create a consistent model representing one floor of a building. It can be ensured that the spaces are modelled by an enclosed boundary, and that the adjacencies of the spaces have been formed correctly.

However the model is incomplete, and therefore to summarise the status of the model, the following points are made:

- a) Only vertical wall constructions have been created, there are no horizontal constructions corresponding to floors and ceilings.
- b) It is implied that the external face of the floor graph was not swept. It would be possible to extrude the external perimeter of the building to create a boundary representation of the external envelope of the whole building. However there is no obvious benefit in creating this representation. The wall constructions on the external boundary would only refer to one face. This fact could be used to distinguish between internal and external constructions, rather than identifying the external representation as a special space.

The modelling process, therefore needs to be extended to complete the model and also to include more building features:

- a) The above process creates a model of one floor, the models of all floors of a building need to be combined into one model, so creating the intermediate floor and ceiling constructions.
- b) Spaces which extend upwards to more than one floor, e.g. stair wells, need to be included.
- c) The modelling method must allow for spaces with non-horizontal ceilings or sloping 'walls', e.g. to represent pitch roofs.
- d) Courtyards will need to be defined, that is external spaces within a floor.
- e) The positioning of windows in walls and roofs will be required.
- f) The thickness of the constructions needs to be modelled in some way, although not important to the topology, the thickness of a wall can have a significant influence on the calculation of the volume of a space.

Therefore, the remainder of the thesis considers the basic modelling processes in more detail, and addresses each of the above requirements.





## 8. CREATION OF PLANAR GRAPH REPRESENTING FLOOR

In Chapter 7, the required Euler Operations were specified to create the planar graph representing a floor of the building. Here the process is described which interprets the geometrical data from the user interface and generates these Euler Operations.

The requirement is for 2-dimensional walls to be defined from which the edges of the planar graph are created. The definition of a 2-dimensional wall is assumed to be purely geometrical, that is it is defined by its two endpoints, and that no topological information is provided or can be derived. Such topological information could be derived if it was assumed that the order in which the walls were input was of significance. For example, if the external perimeter of the building was defined by an ordered set of walls, this would infer the connectivity between adjacent walls. However, as it is not desirable to impose such constraints on the user, no such assumption is made, and the walls can be input in an entirely random order.

Also, it cannot be assumed that there is a one-to-one relationship between walls and edges. An edge is adjacent to only two faces, whereas a length of wall may be input which is split into several edges, dependent on other walls connecting to it, or intersecting with it. See figure 8-1.

Therefore, the required process can be generalised into one of creating a planar graph from 2-dimensional line segments and is summarised:

- a) The input is the 2-dimensional endpoints of a line segment
- b) Dependent on this geometry the appropriate Euler Operations are generated to create the derived edge(s) and other topology
- c) The appropriate geometry is attached to the topology.

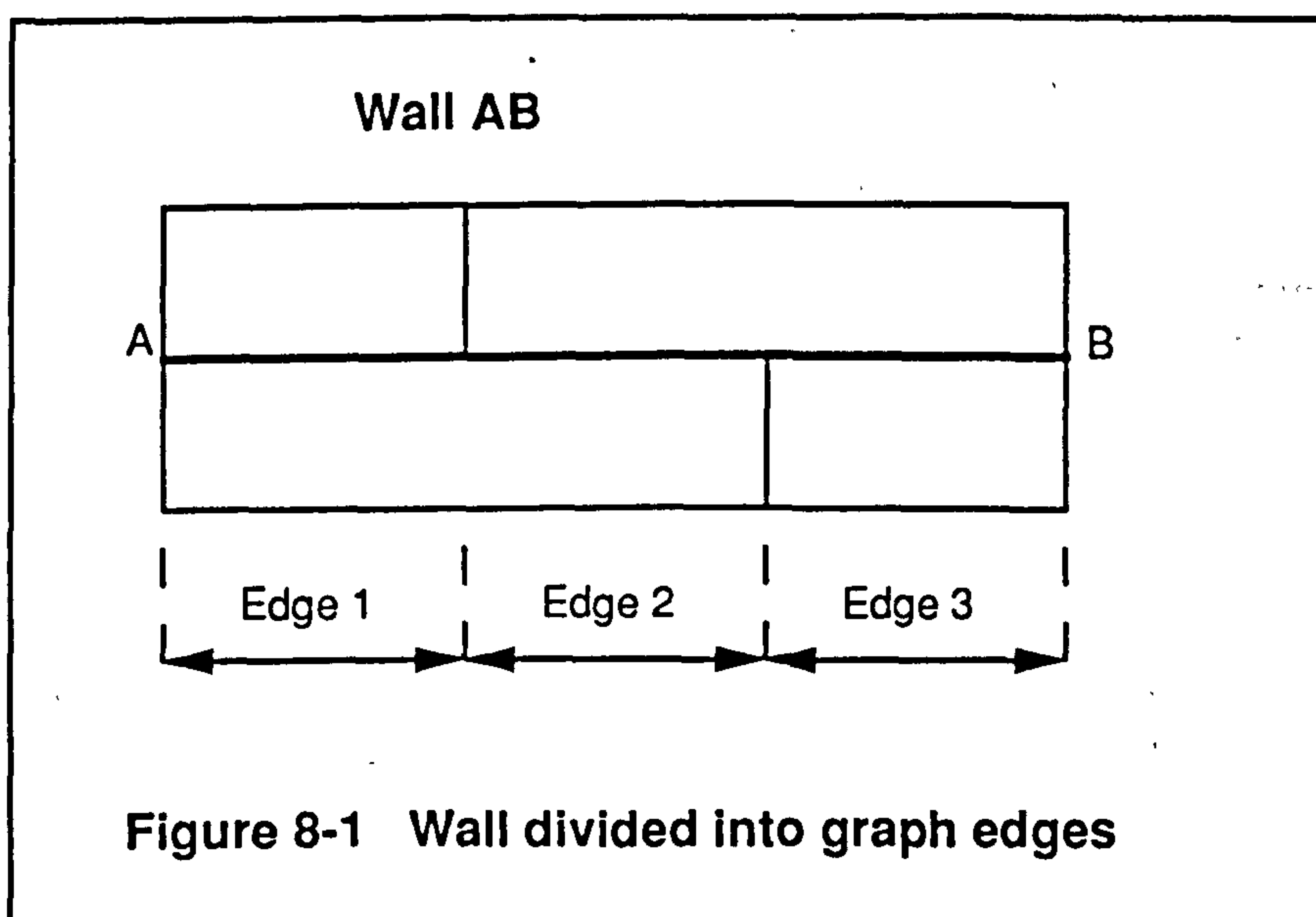
### 8.1 Specification of Process

The process is defined by considering the general case of an existing planar graph, for which new edges are to be generated from an input line segment. The endpoints of the line may be geometrically positioned anywhere within the planar graph, and the line may intersect any number of faces or edges of the graph.

The major steps of the process are:

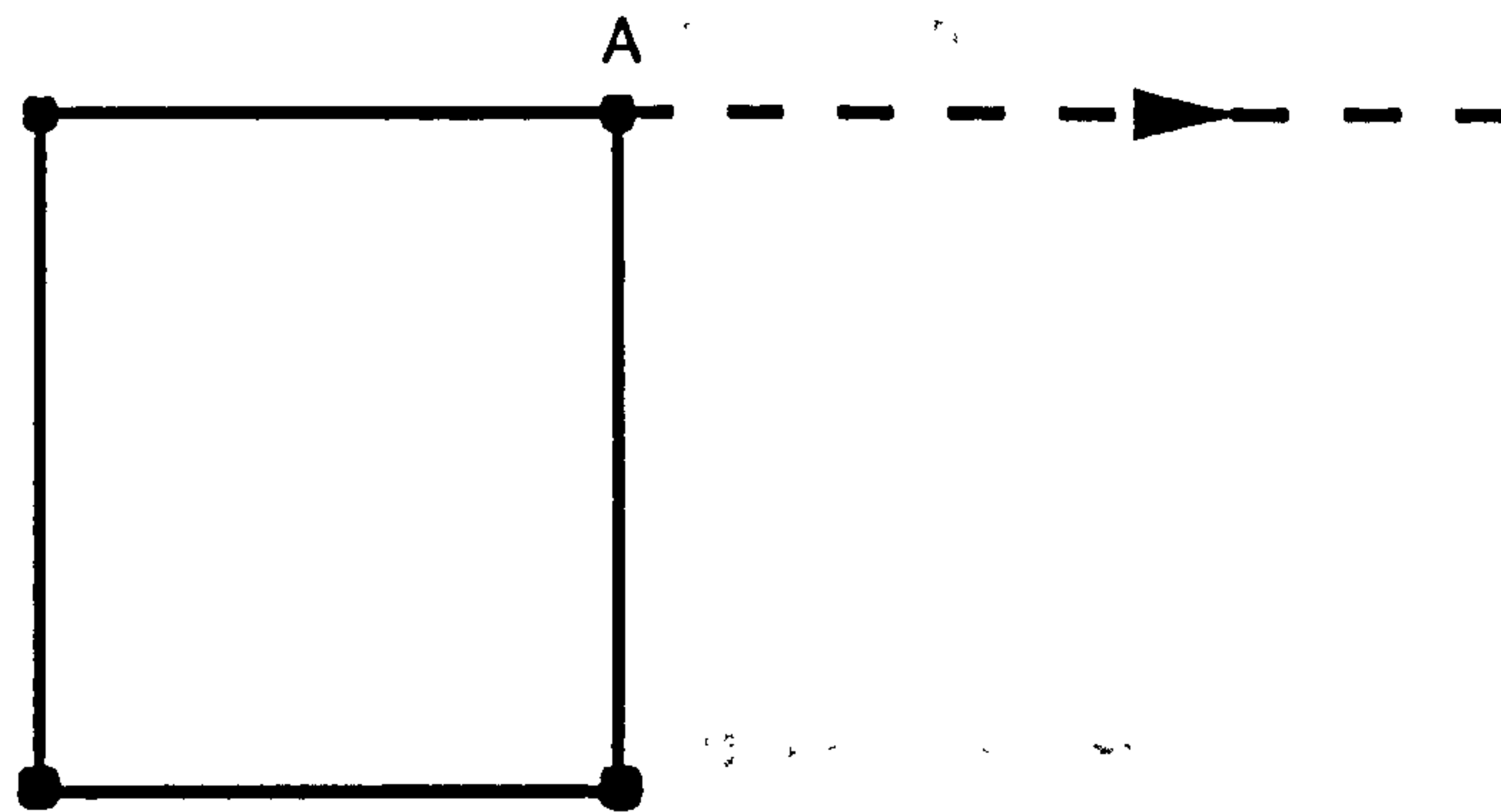
1. Find location of the first endpoint and generate the appropriate Euler operation to create a vertex. One of the following cases will apply, (illustrated in figure 8-2):

- i) This is the first vertex of the graph : 'Make a body, face and vertex'
- ii) Point is co-incident with an existing vertex : No Euler operation is necessary.



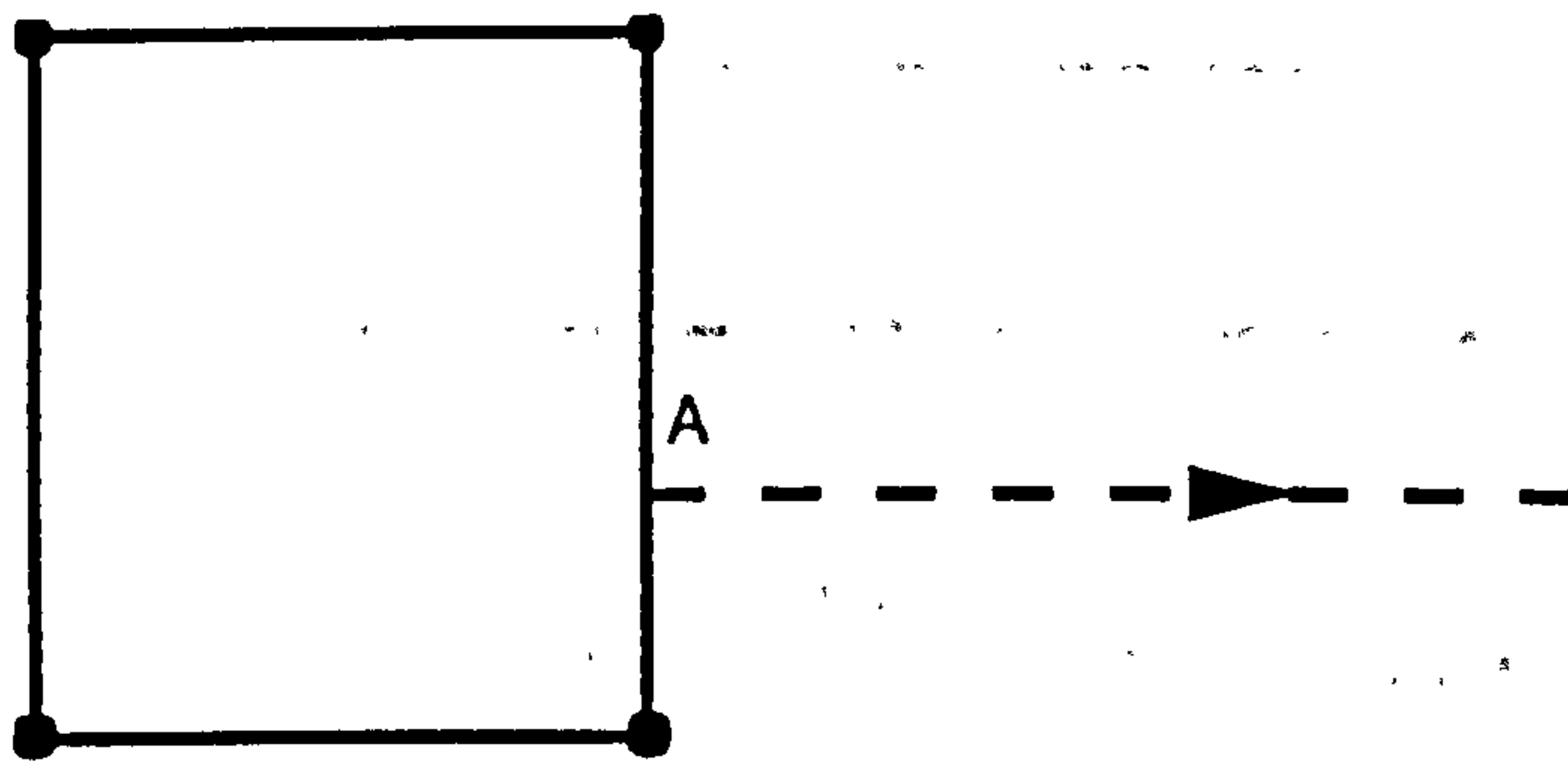
A ———▶

1i) Endpoint is first of graph:  
Make a body, face and vertex at A

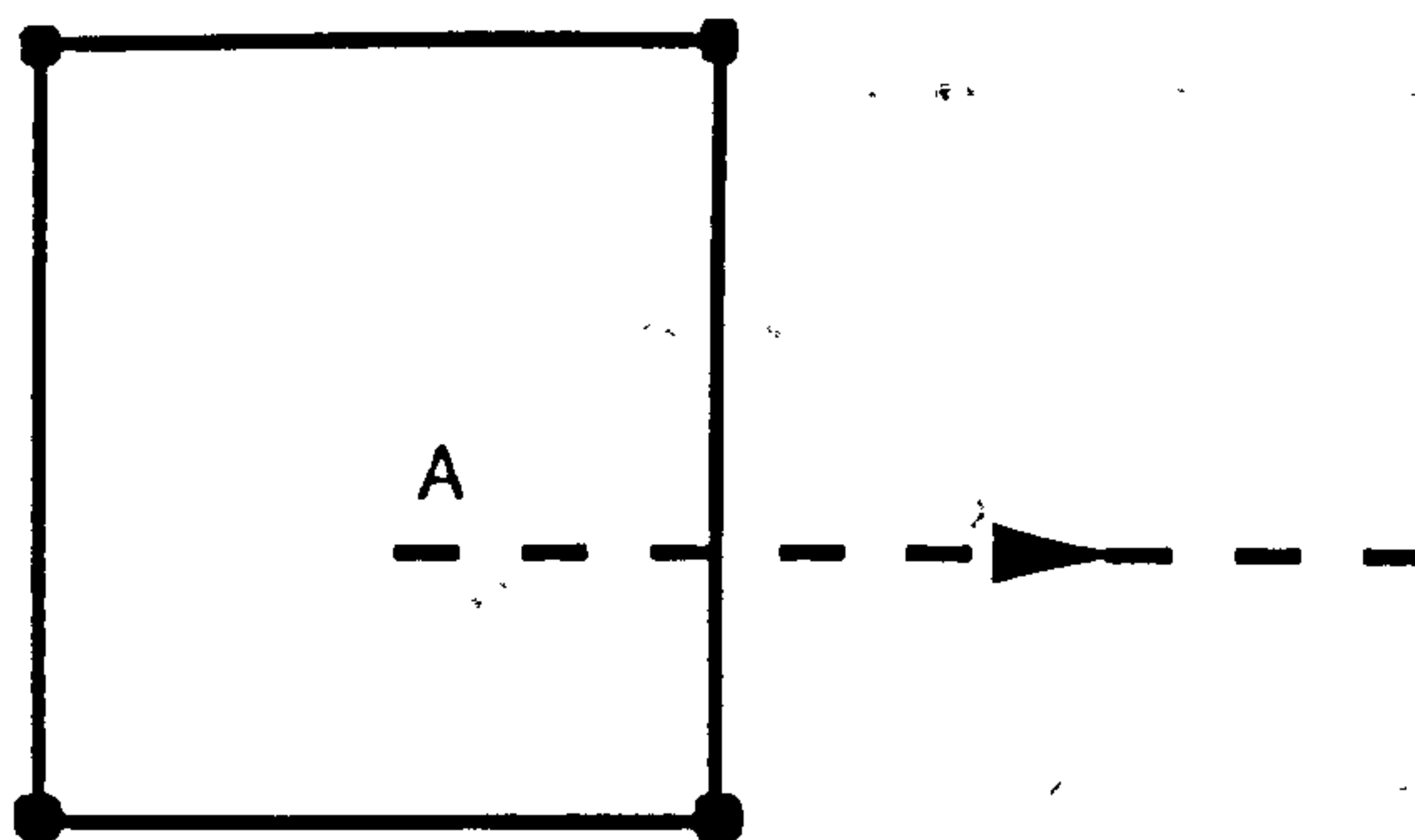


1ii) Endpoint is coincident with  
existing vertex A - no Euler operation

**Figure 8-2 Creation of vertex for 1st endpoint of wall**



1iii) Endpoint is on an edge:  
Split-edge to create vertex at A



1iv) Endpoint is within a face:  
Make vertex hole loop at A

Figure 8-2 (Continued) Creation of vertex for 1st  
endpoint of wall



iii) Point is located on an edge, but not at the vertex of the edge:  
'Split-edge'

iv) Point is within a face, possibly the graph external face:  
'Make a vertex, hole loop'

Therefore the vertex at the start of the new edge is specified.

2. Find the loop and hence face, adjacent to the vertex, into which the line segment is projecting from the vertex, dependent on the direction of the line segment.

For cases i) and iv) above, the face is already defined.

3. Find any intersections of the line segment with the edges or vertices of the face and generate the appropriate Euler operation to create an edge. More than one intersection is possible if the face is concave in shape.

Either no intersection is found (figure 8-3):

'Make an edge and vertex'.

Or intersection(s) found:

i) Find intersection nearest to start point.

ii) If the intersection is on an edge: 'Split-edge', to create next vertex, otherwise intersection is at a vertex which becomes the next vertex. (Figure 8-4)

iii) Get loop of next vertex relating to face.

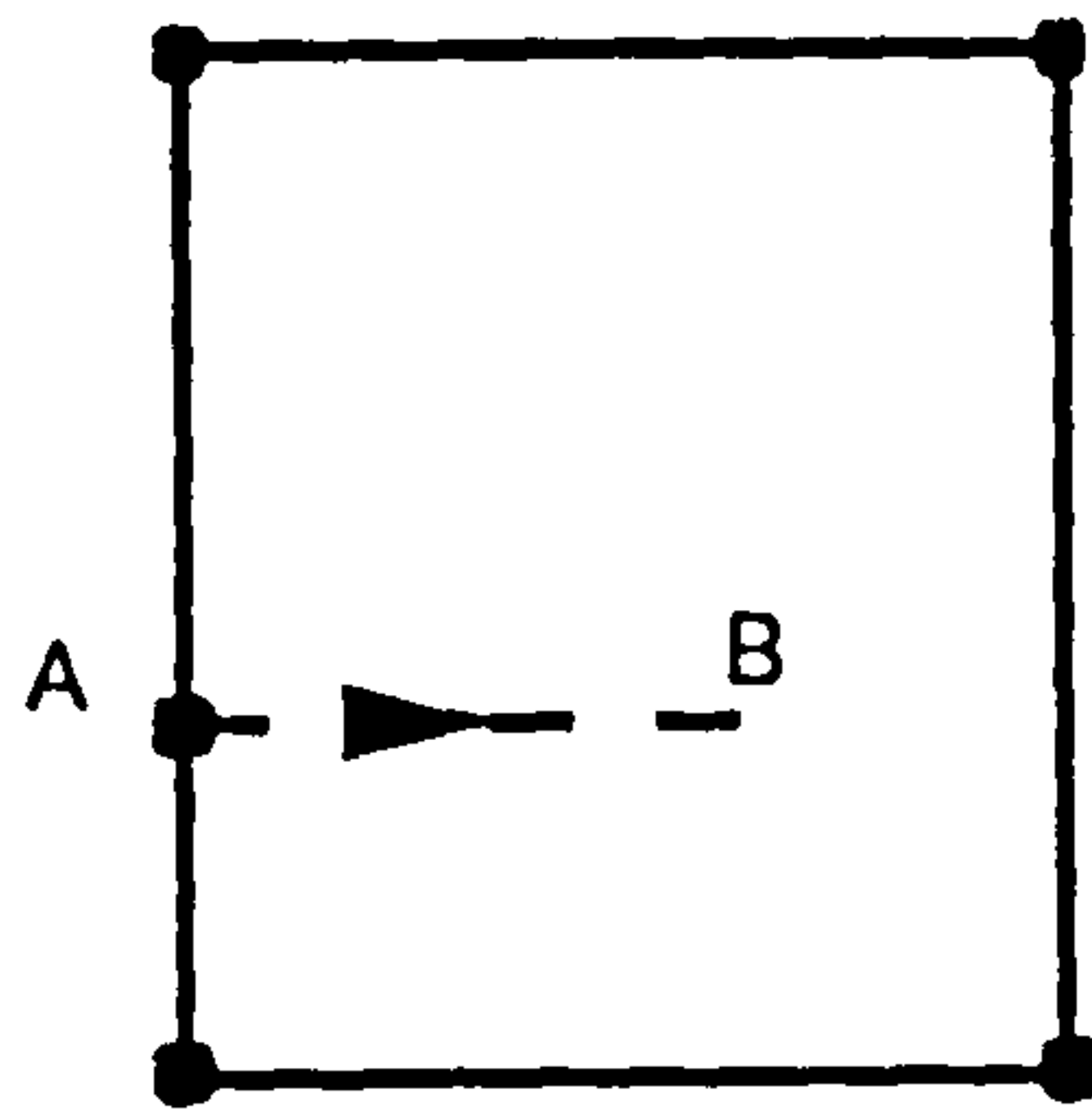
iv) Either loops of starting vertex and loop of next vertex are identical : 'Make an edge and face'. (Figure 8-5)

Or loops differ: 'Make an edge and delete a hole loop'. (Figure 8-6)

4. If an intersection is found in 3, then all of the line segment may not have been processed and more edges need to be created. 'Next vertex' in 3 becomes starting vertex, and steps 2 to 4 are repeated. Until either no intersection is found in 3, or an intersection is found whose position co-incides with the endpoint of the line.

### 8.1.1 Checking for Coincident Edge

As it has been assumed that any 2D line can be 'drawn' across the graph, it is possible that it may be completely coincident or partially coincident with an existing edge or edges. Although the occurrence of coincident edges would most probably be avoided by a user placing 2D lines representing walls, this cannot be assumed and the above process is modified to allow for this. See figure 8-7 for examples.



**Figure 8-3 No intersection found: Make edge, and vertex**

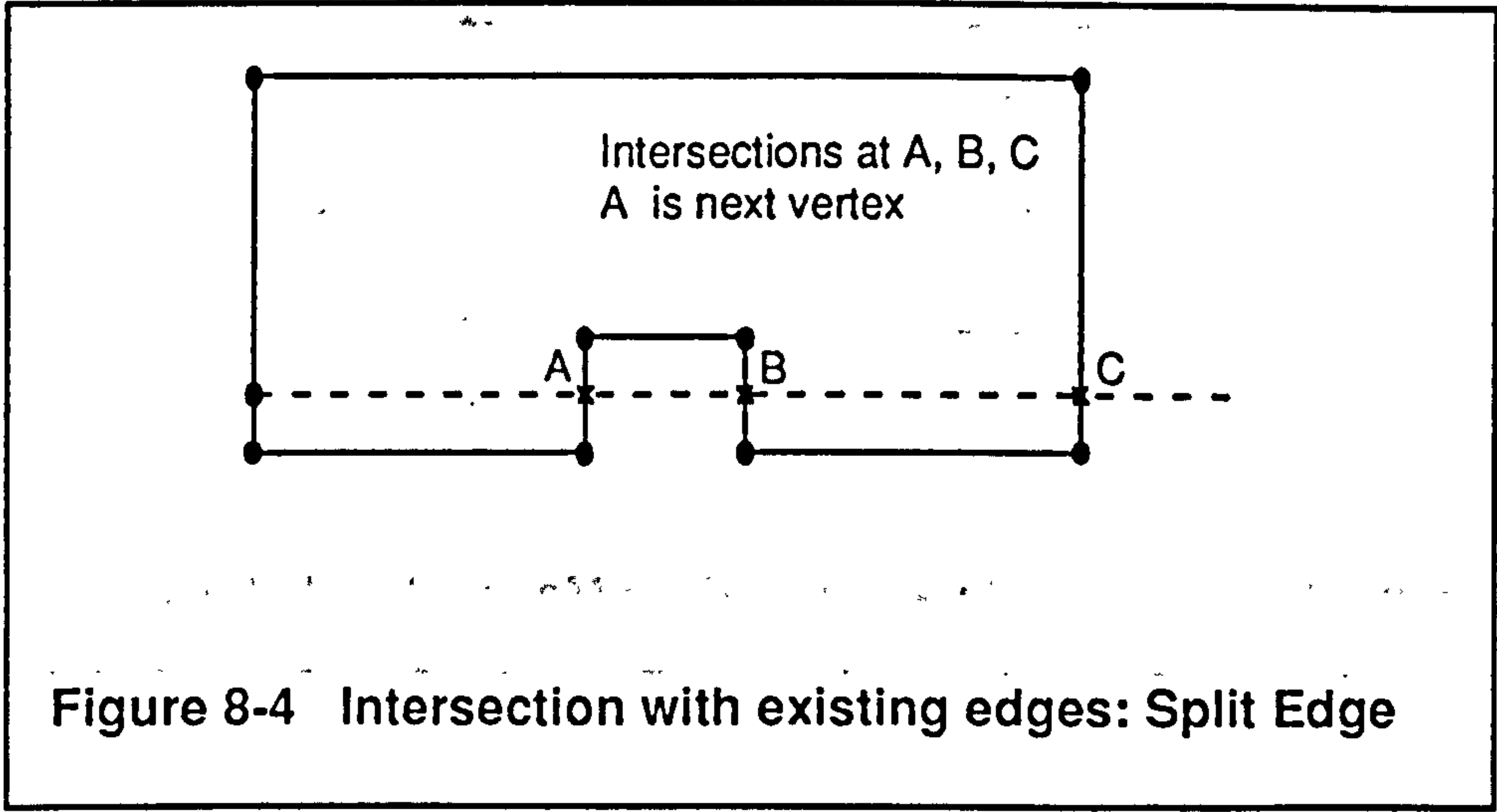


Figure 8-4 Intersection with existing edges: Split Edge

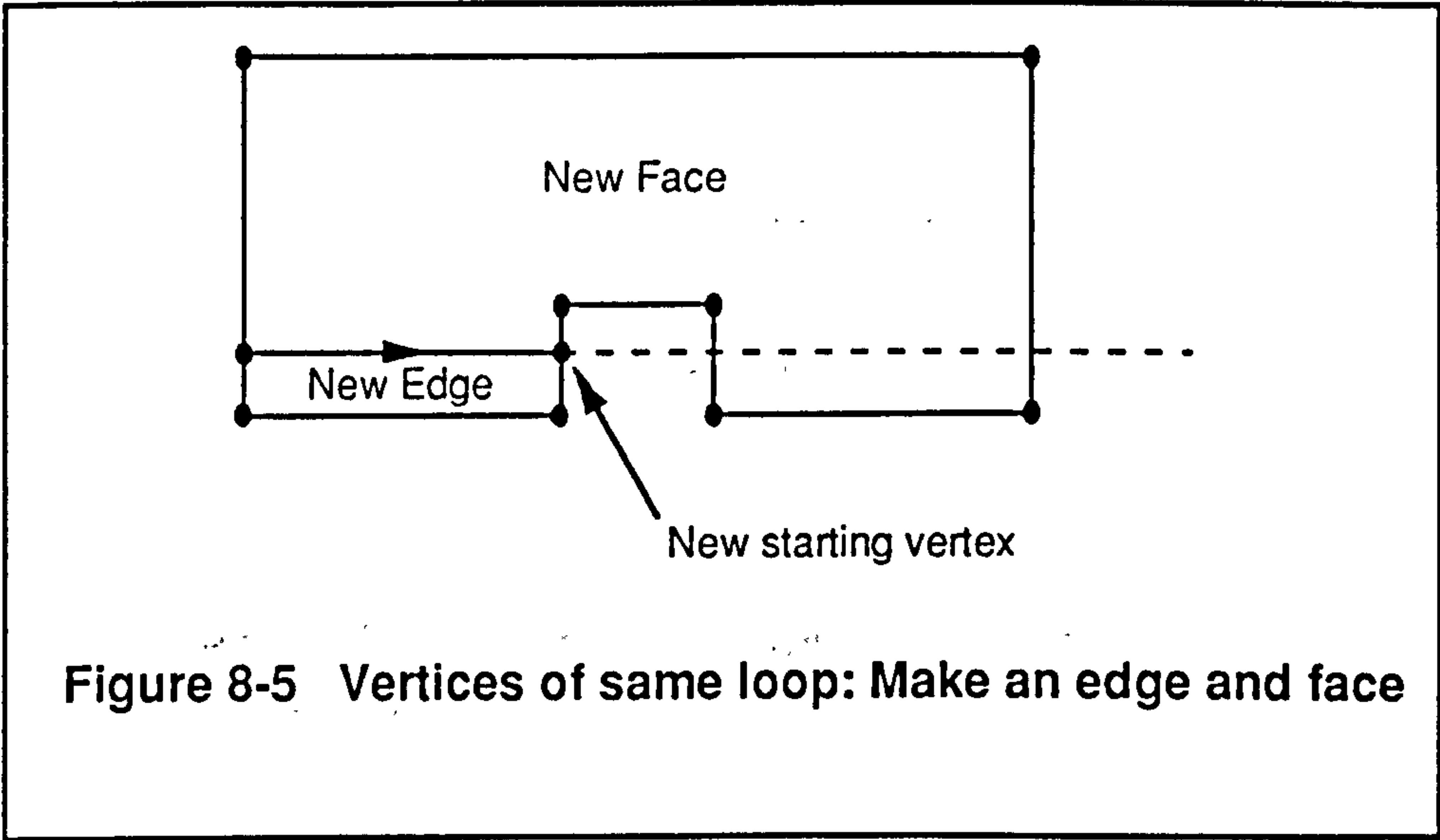


Figure 8-5 Vertices of same loop: Make an edge and face

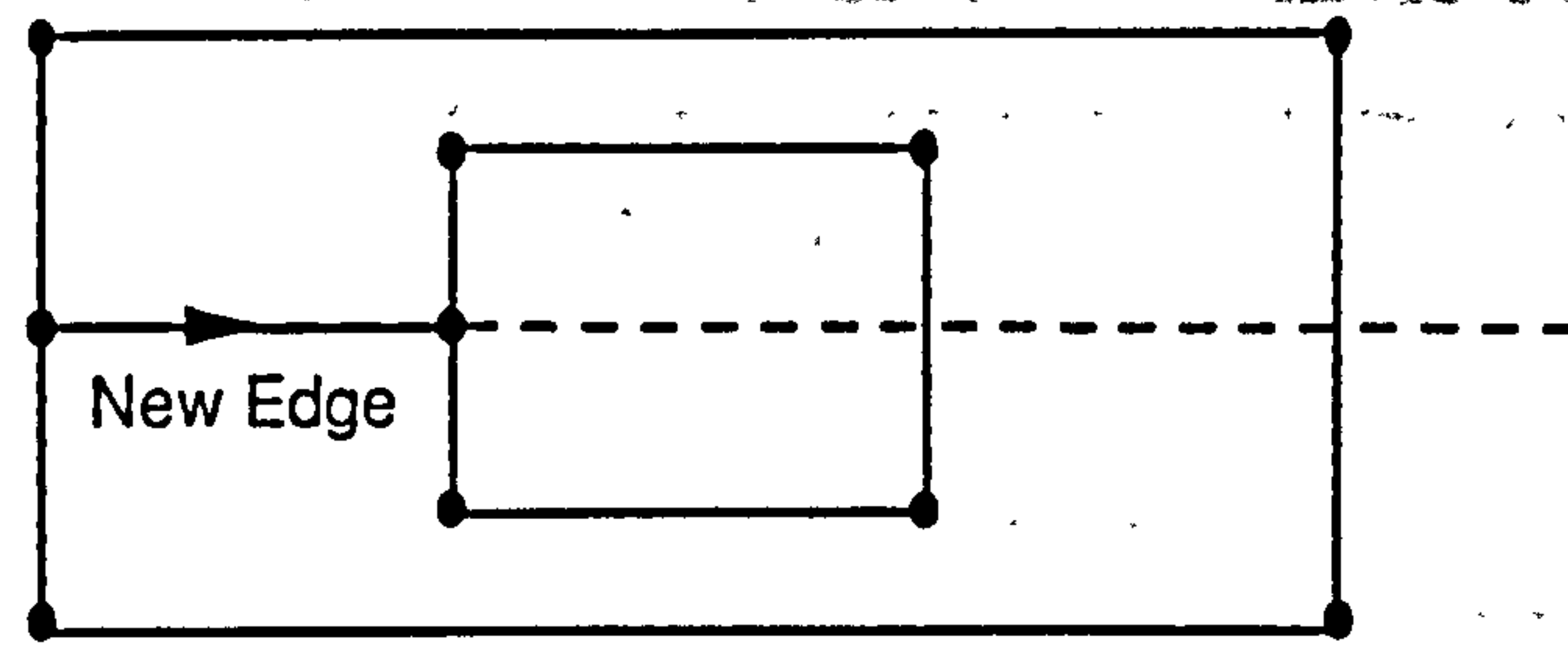


Figure 8-6 Loops of vertices differ: Make edge, kill a loop

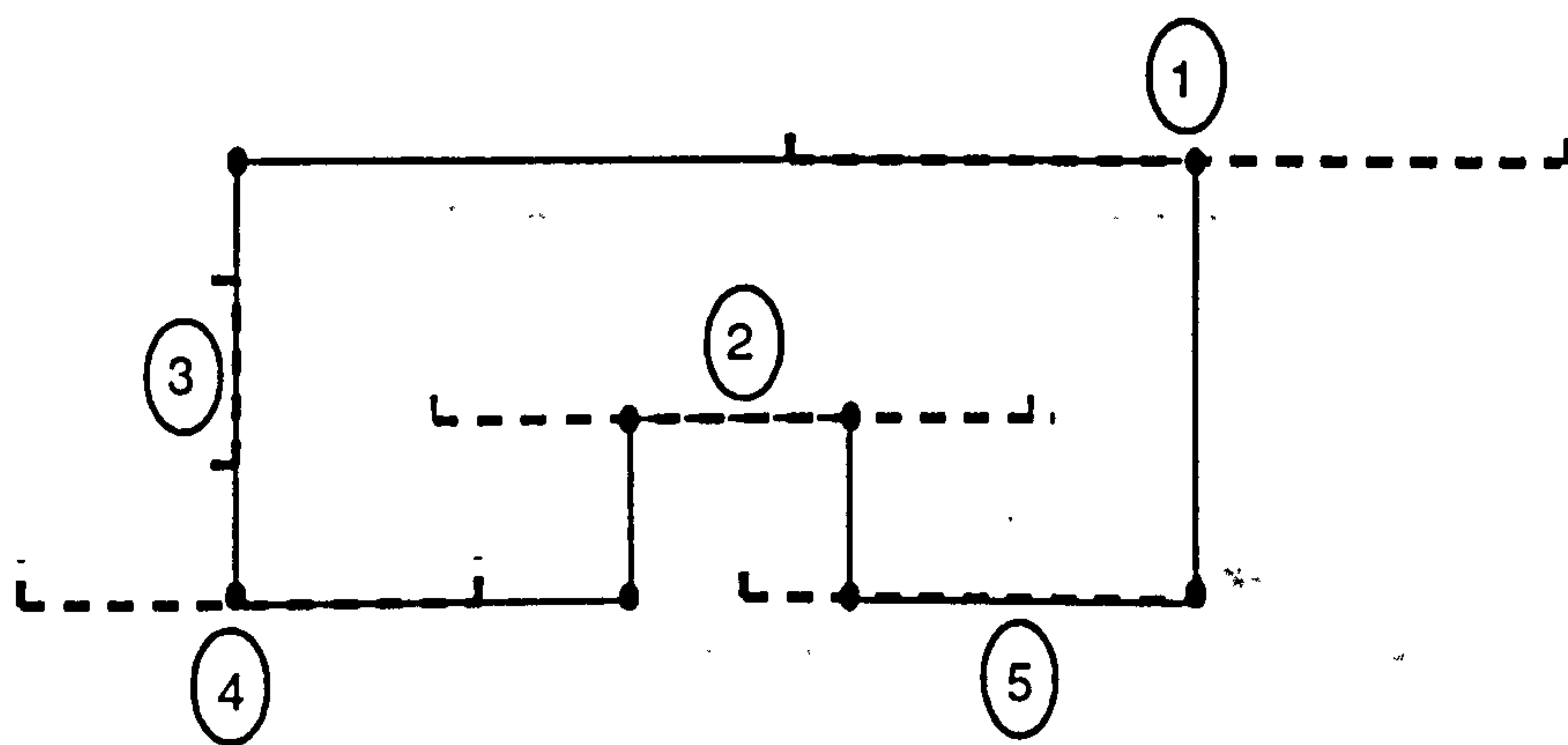


Figure 8-7 Examples of coincident edges



Therefore for each vertex at the start of an edge, which can be either the first endpoint of the line, or a subsequent starting vertex, the edges of the vertex have to be checked for 'coincidence'. A coincident edge has the same direction as the line.

If there is a coincident edge, and the endpoint of the line is on the edge, or is coincident with the other vertex of the edge, then no more edges need to be added, as in examples 3, 4 and 5 of figure 8-7.

Otherwise, the line must extend further than the other vertex of the edge, and therefore this vertex becomes the next starting vertex instead, as in examples 1 and 2 of figure 8-7. Therefore this check for coincident edges is made at the start of step 2.

## 8.2 Attachment of Edges

As detailed in Chapter 5, the Euler operations require the specification of how an edge is attached to each vertex. No closed edges are possible given the geometry of the planar graph. Therefore the attachment of an edge is completely specified by the next edge clockwise around the vertex.

The clockwise edge is found by the following process, given the direction of the line segment (see figure 8-8):

1. Get the edges attached to a vertex

For each edge:

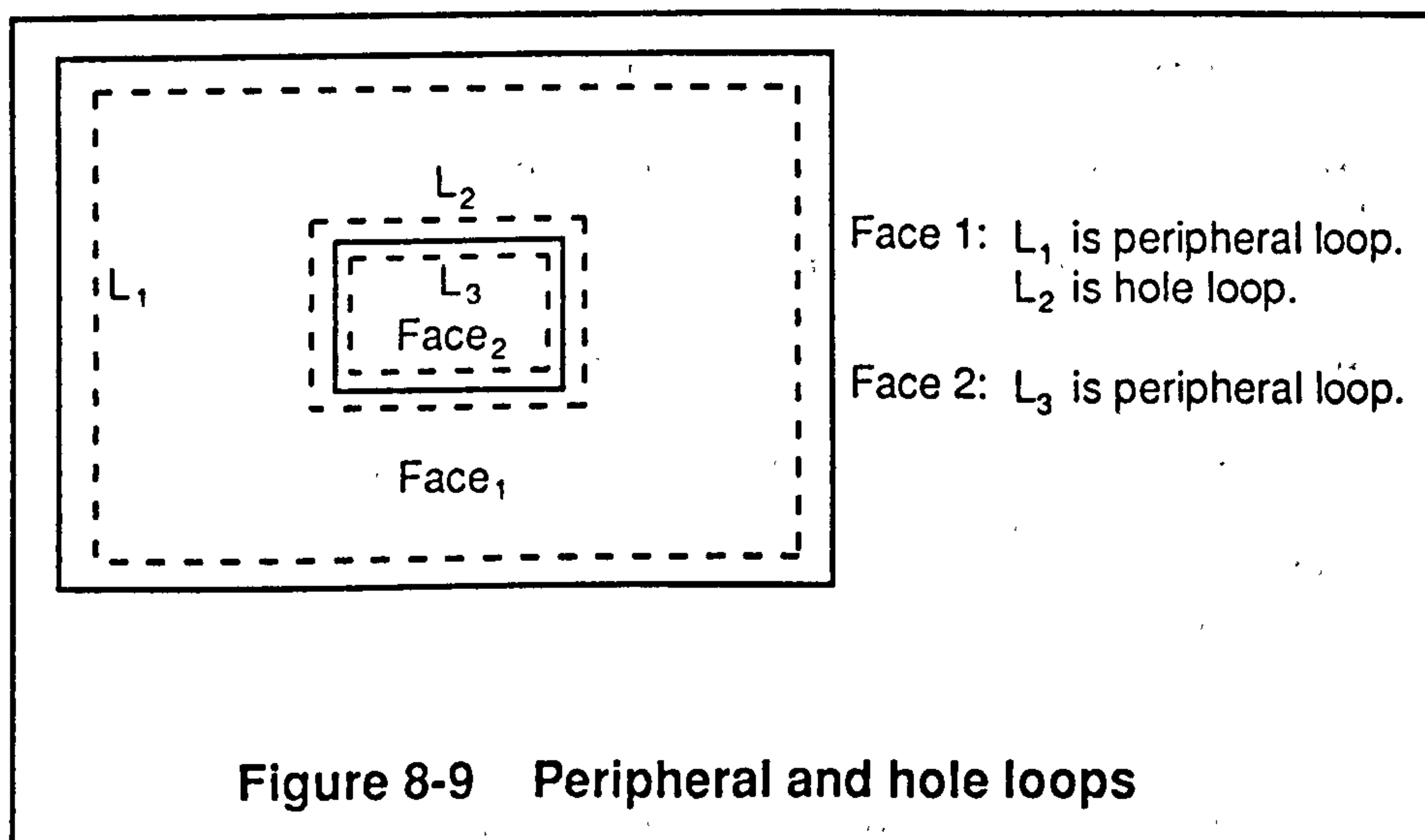
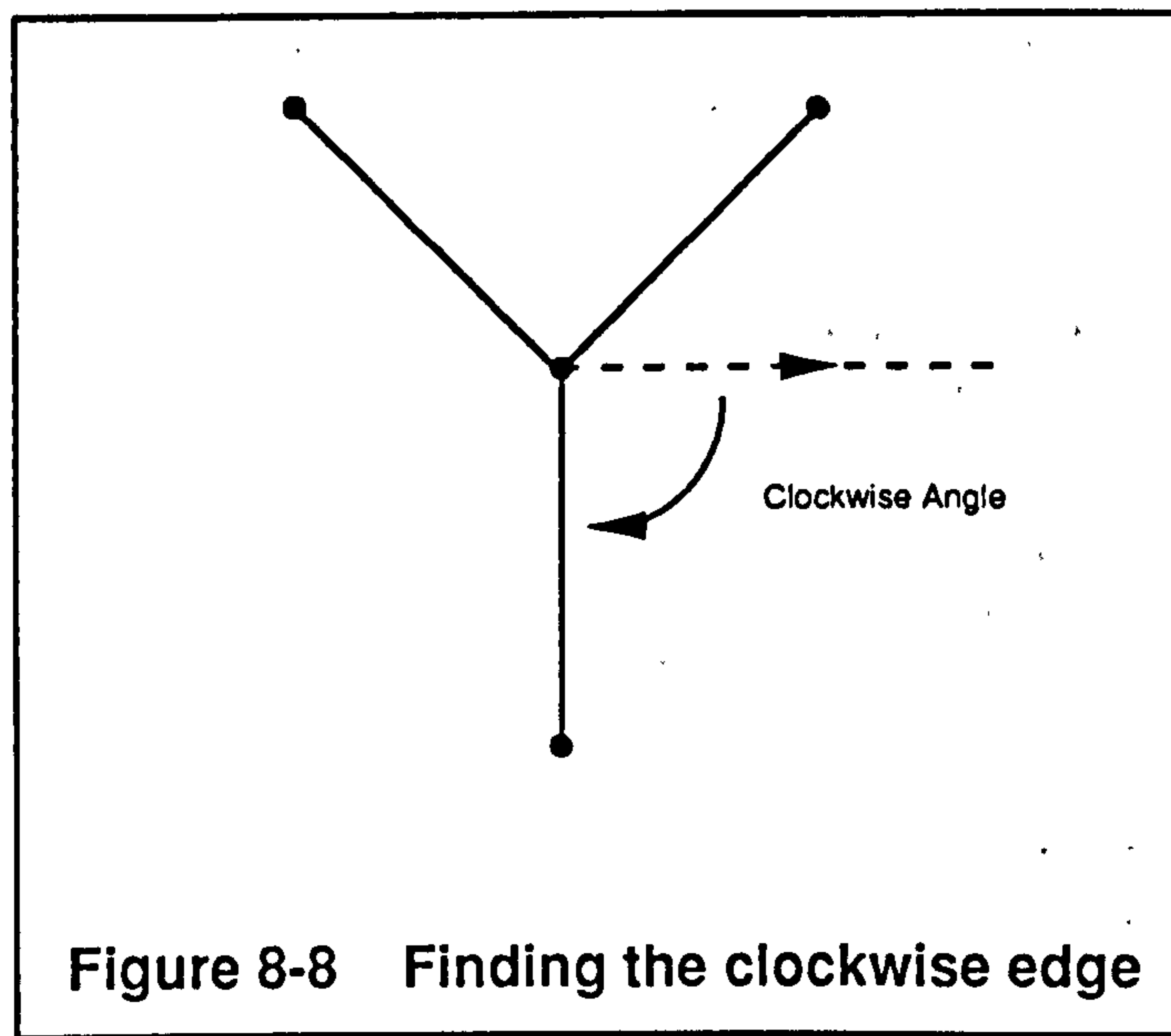
2. Calculate direction vector of edge away from vertex
3. Calculate clockwise angle between line and edge vector.
4. The required edge is that corresponding to the smallest angle.

The above process is also used to determine the loop and hence the face in step 2 of the process 8.1. The required loop is the loop of the edge found as above, such that the edge is counter-clockwise from the vertex in the loop.

Note that even when specifying the attachment of the second vertex when the loop of the vertex may be known, the topological access function suggested in Chapter 5: 'edge counter-clockwise in loop from vertex' cannot be assumed to give the required edge, as, if there are wire-edges at the vertex in the loop, more than 1 edge satisfies the topological condition.

## 8.3 Faces and Loops

Each face may have more than one loop of edges, a peripheral loop and any number of hole loops contained within it. The external face is an exception to this, as it has only hole loops, there is not one loop of edges entirely enclosing the other loops. This definition of peripheral and hole loops is determined by the geometry of the edges of the loops. See figure 8-9.



It is necessary to be able to identify the external face of the graph, and to ensure that it only has hole loops, and also to be able to identify the peripheral loop of each other face, as distinct from their hole loops. This is required for the sweep operation described later, but also for the creation of the corresponding graphical display, when operations require the identification of a space.

Therefore, when adding an edge to create a face or delete a hole loop, the data structure is kept constant in that:

- a) the first face of the winged-edge structure in the faces list is always the external face.
- b) the first loop of a face is always the peripheral loop.

These are conventions only relevant to this representation within a winged-edge data structure, and is dependent on the geometry. To ensure this, the allocation of loops must be considered for the above operations: 'Make an edge and a face', and 'Make an edge, kill a loop'.

### 8.3.1 Creation of a Face

The implementation of the operation 'Make an edge and a face' is such that the new face is created to the left of the new edge, i.e. the original face, divided into two parts by the edge, is to the right of the new edge.

This operation always joins two vertices of the same loop. There is a requirement to provide the facility to move the loops of the original face to that of the new face as part of this Euler operation (See Chapter 5). To ensure the conventions above are always valid, the following cases of moving loops between faces are allowed for:

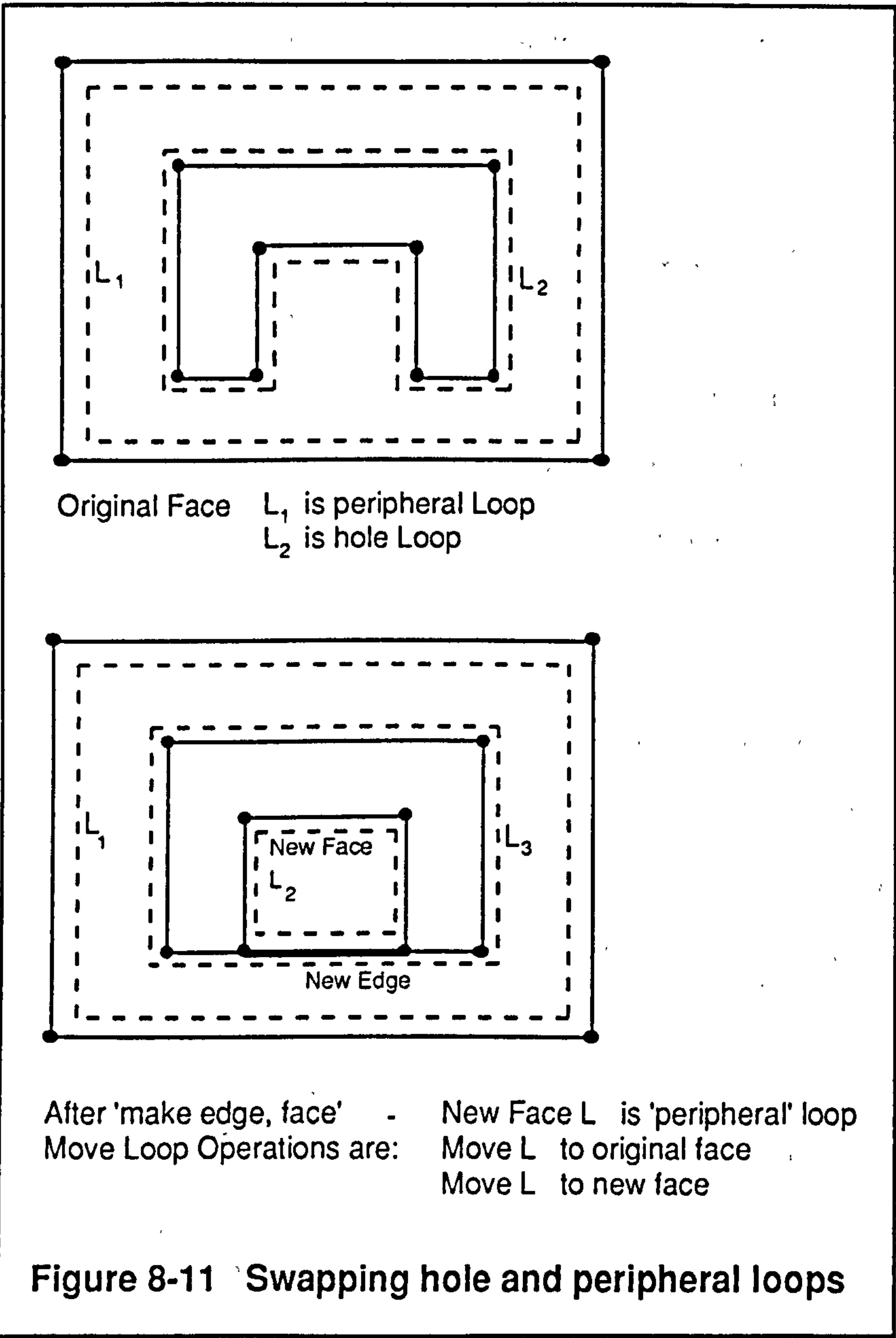
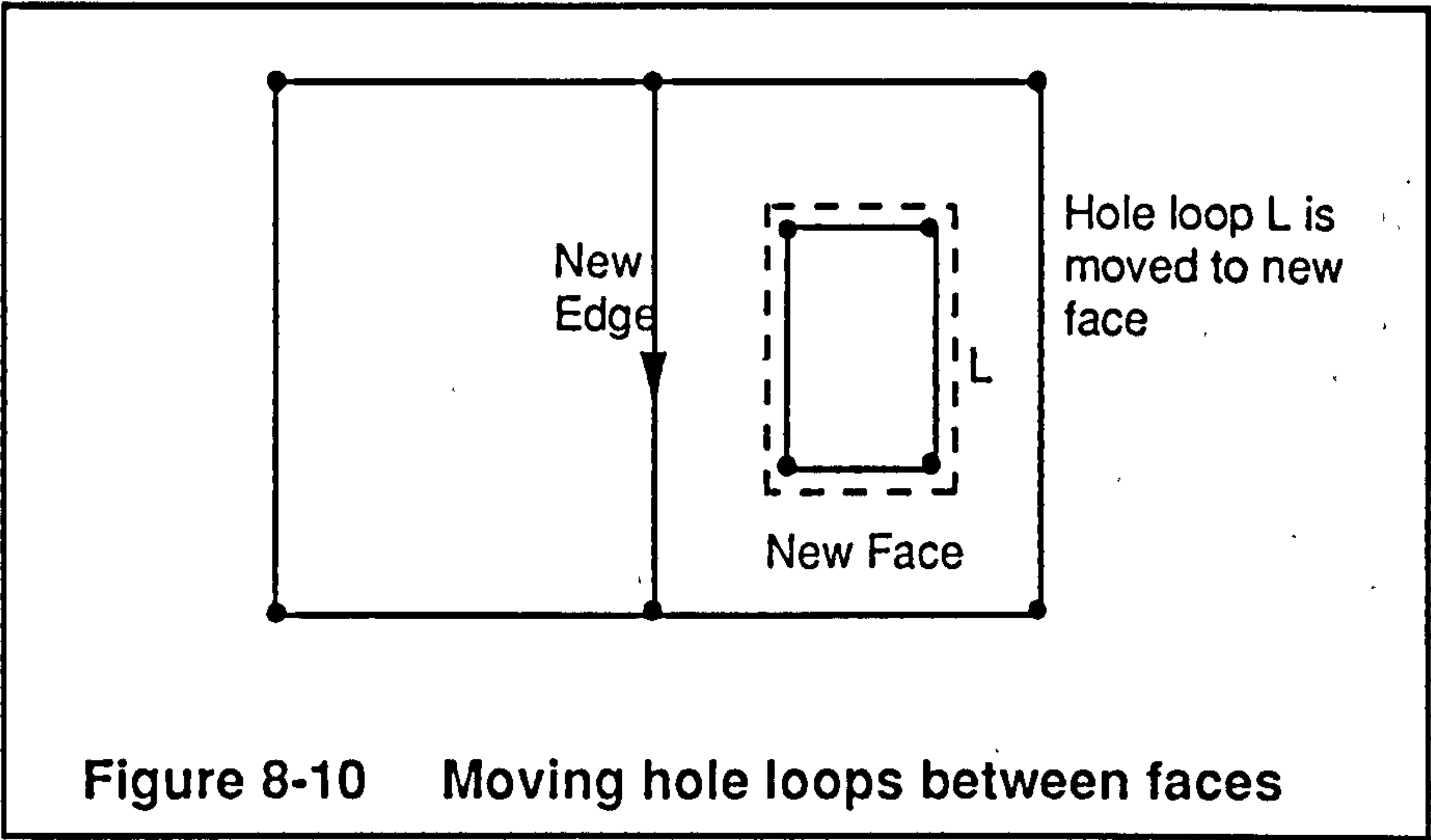
- a) Moving hole loops between faces:

This is the simplest case where the division of a face has resulted in the new face containing some of the hole loops of the original face. See figure 8-10.

- b) Swapping hole and peripheral loops:

When joining two vertices of a hole loop, it must be ensured that this hole loop remains a loop of the original face, and that a new peripheral loop is created for the new face.

Therefore the new loop, created and defined as the only loop of the new face, is checked as to whether it is a hole loop. If it is, it is moved back to the original face, and the original modified loop that was joined is moved to the new face. See figure 8-11. This is dependent on the fact that when creating a new face, at least one of the loops on either side of the new edge must be a peripheral loop.





c) Special case of External Face:

It is worth noting that b) above may also apply when the external face is 'divided', except that a hole loop may be moved back to the original face and be the only loop of the face if that face is the external face. See figure 8-12.

### 8.3.2 Deletion of a Hole Loop

This operation joins vertices of different loops of the same face. Therefore a loop is deleted by joining two loops into one loop. The possible cases are:

a) Joining peripheral loop to hole loop (figure 8-13):

The resulting loop must be the peripheral loop of the face. This can easily be ensured by checking whether either of the original loops is the first loop referenced by the face, if so the resulting loop must then be this first loop.

b) Joining two hole loops (figure 8-14):

As indicated above, this case can be established by checking that neither of the loops are the first loop referenced by the face. The resulting loop is a hole loop, therefore no further checks are necessary.

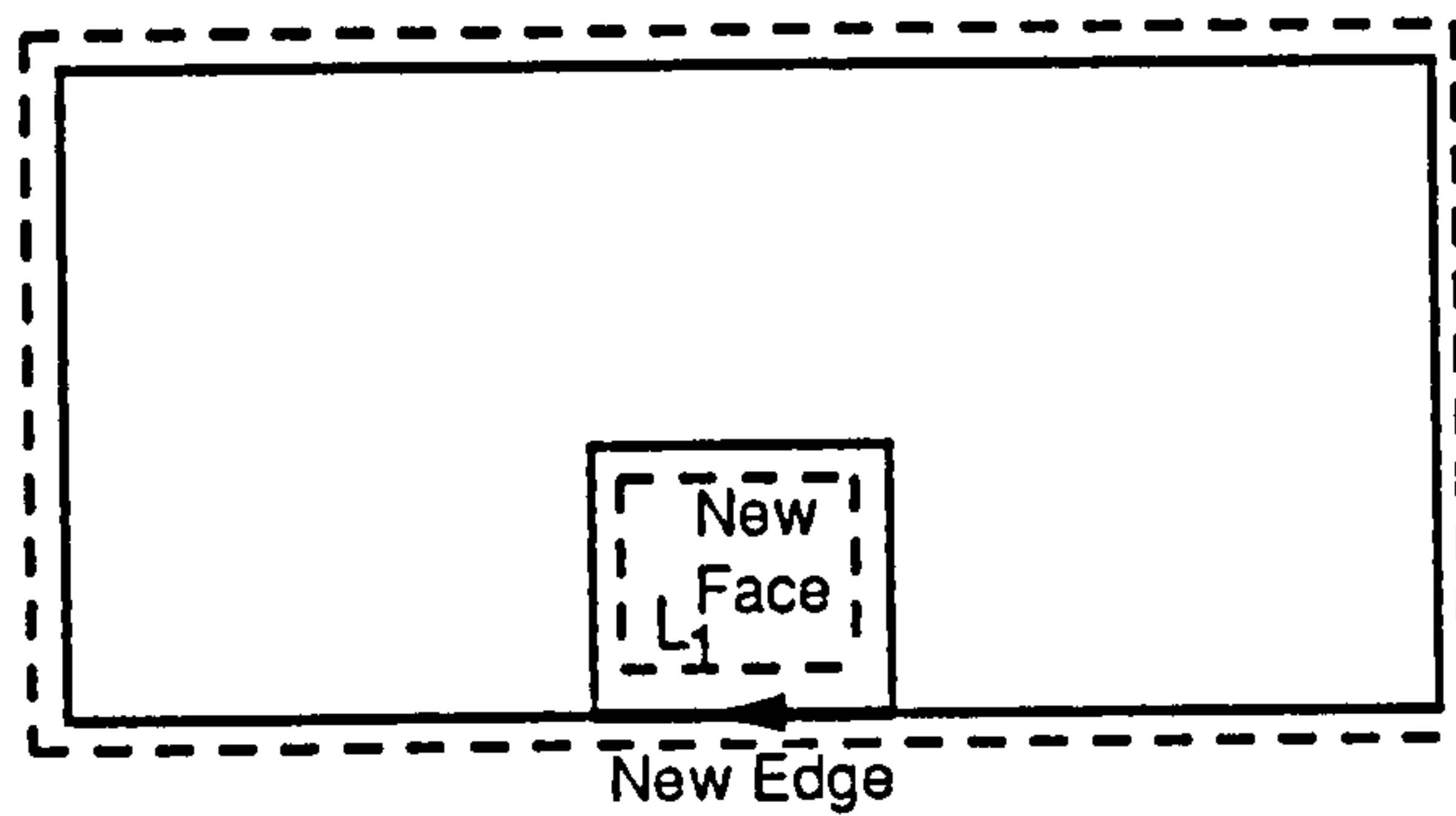
Therefore, no geometrical checks are required for this operation.

### 8.3.3 Geometrical Calculations for Loops and Faces

From the above, it can be seen that there are some geometrical calculations required, concerning loops and faces. There are basically two calculations.

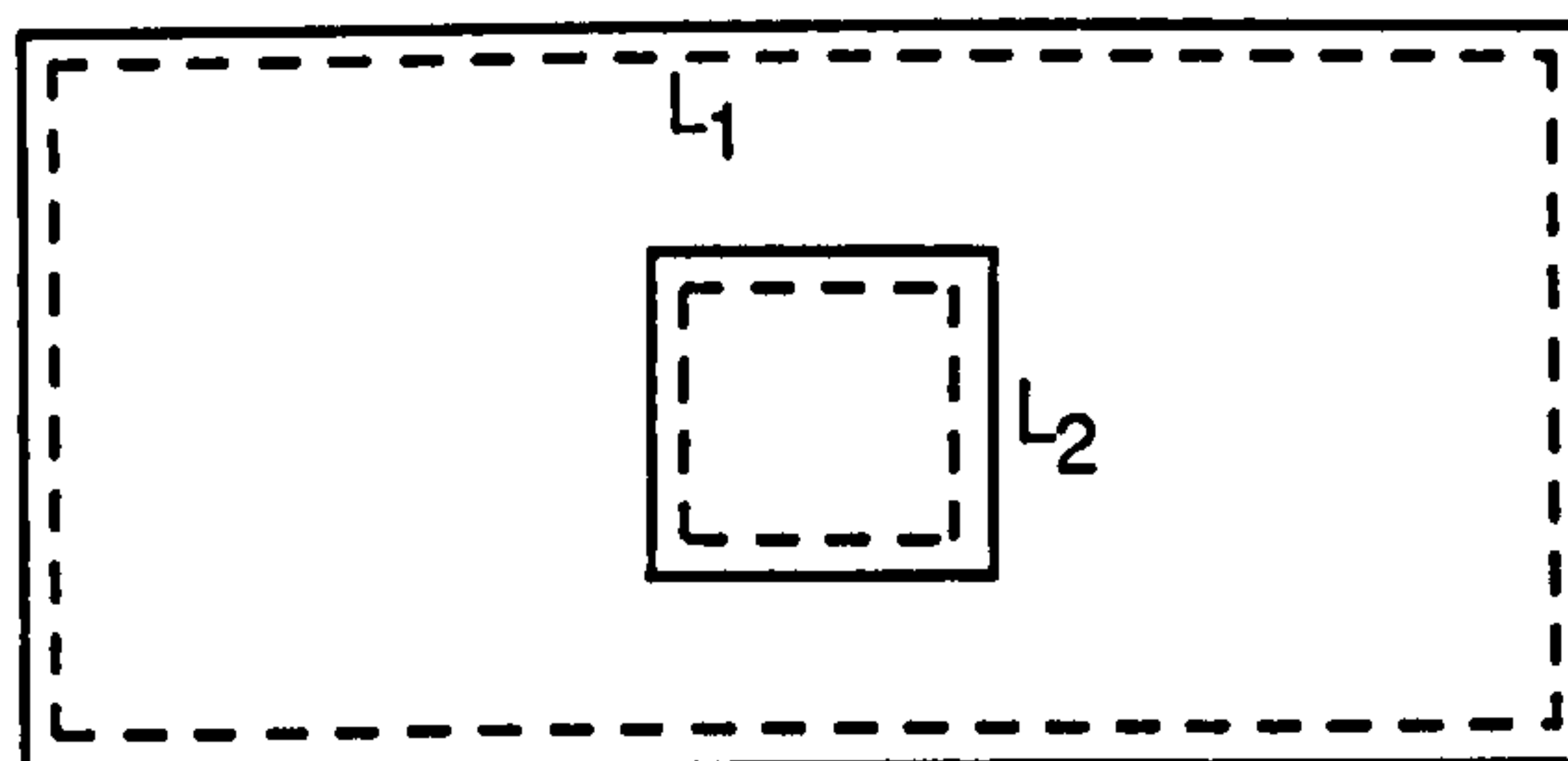
a) Establishing peripheral and hole loops:

By calculating the sum of the angles at each vertex of a loop, see figure 8-15, the total is either  $2\pi$  or  $-2\pi$ . When the edges are accessed in a clockwise direction in relation to the loop, and the counter-clockwise angle between the 2 vectors at each vertex is calculated within the range  $-\pi$  to  $\pi$ , the sum of these angles for a peripheral loop is  $-2\pi$ , for a hole loop, it is  $2\pi$ . This is required to distinguish between peripheral and hole loops as required in 8.3.1 b) above.

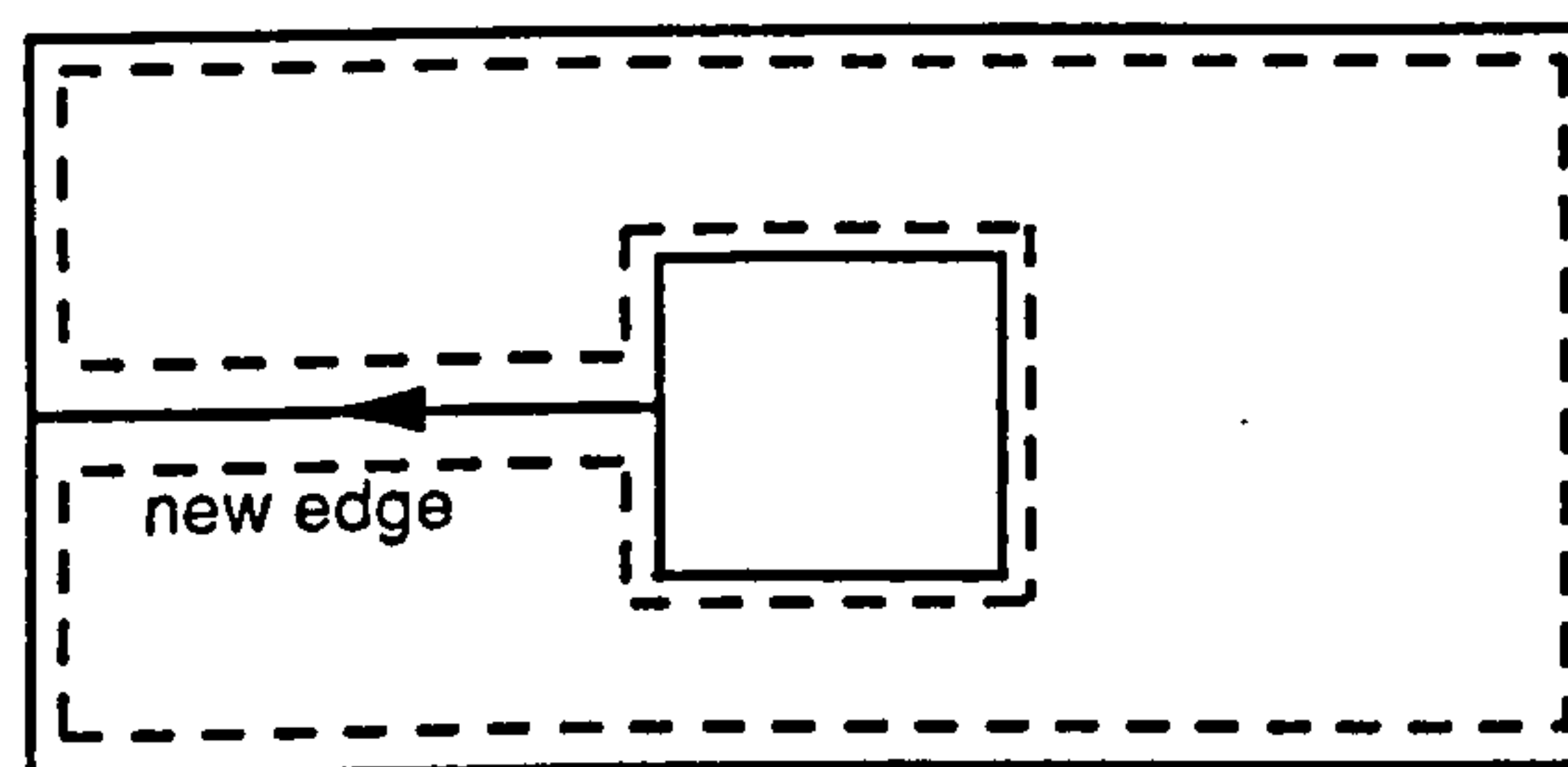


L1 is original loop of external face.  
 After 'make edge, face':  
 Move L2 to new internal face.  
 Move L1 to external face.

**Figure 8-12 Swapping hole and peripheral loops of external face**

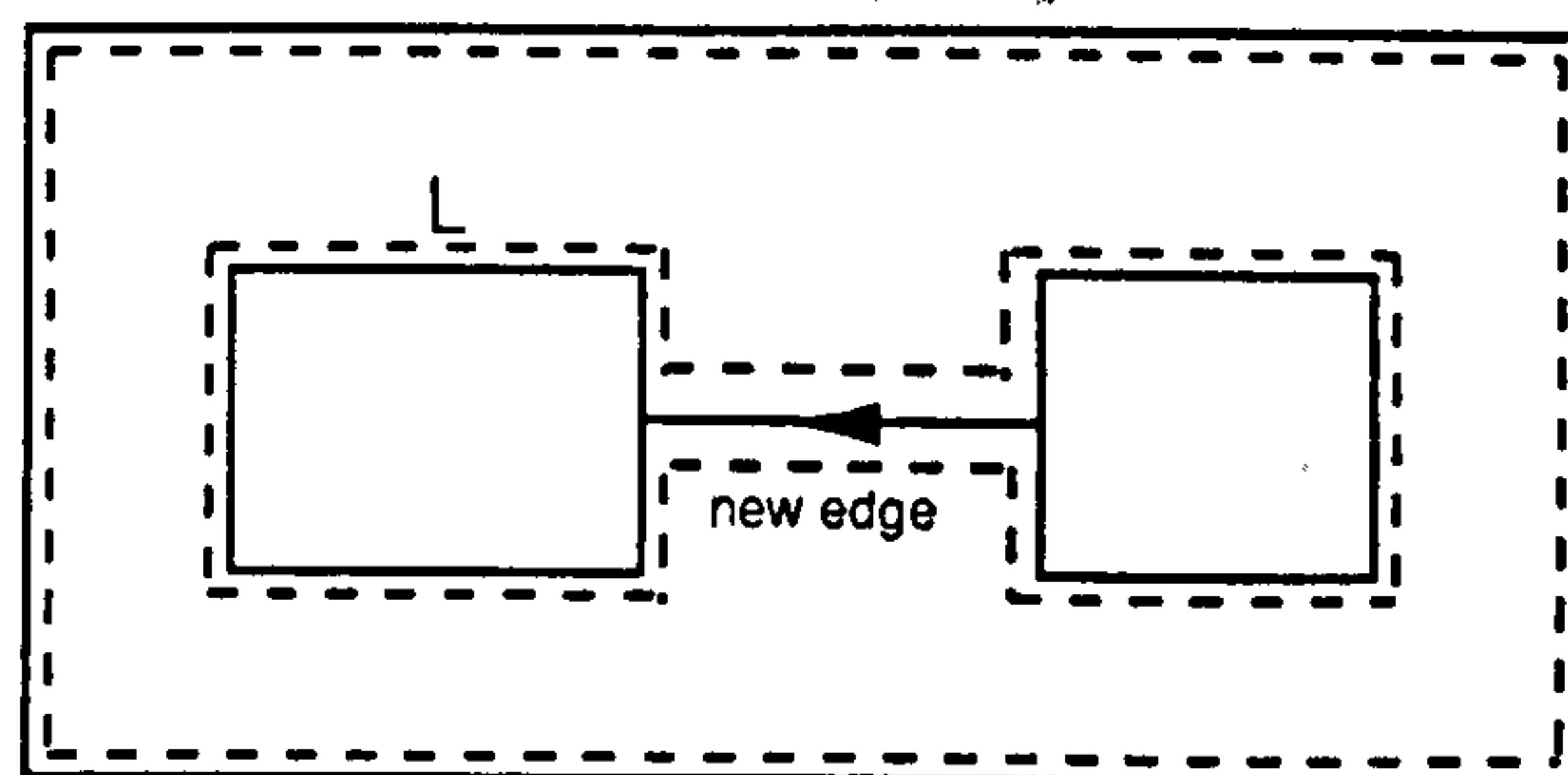


L1 and L2 are loops of face.



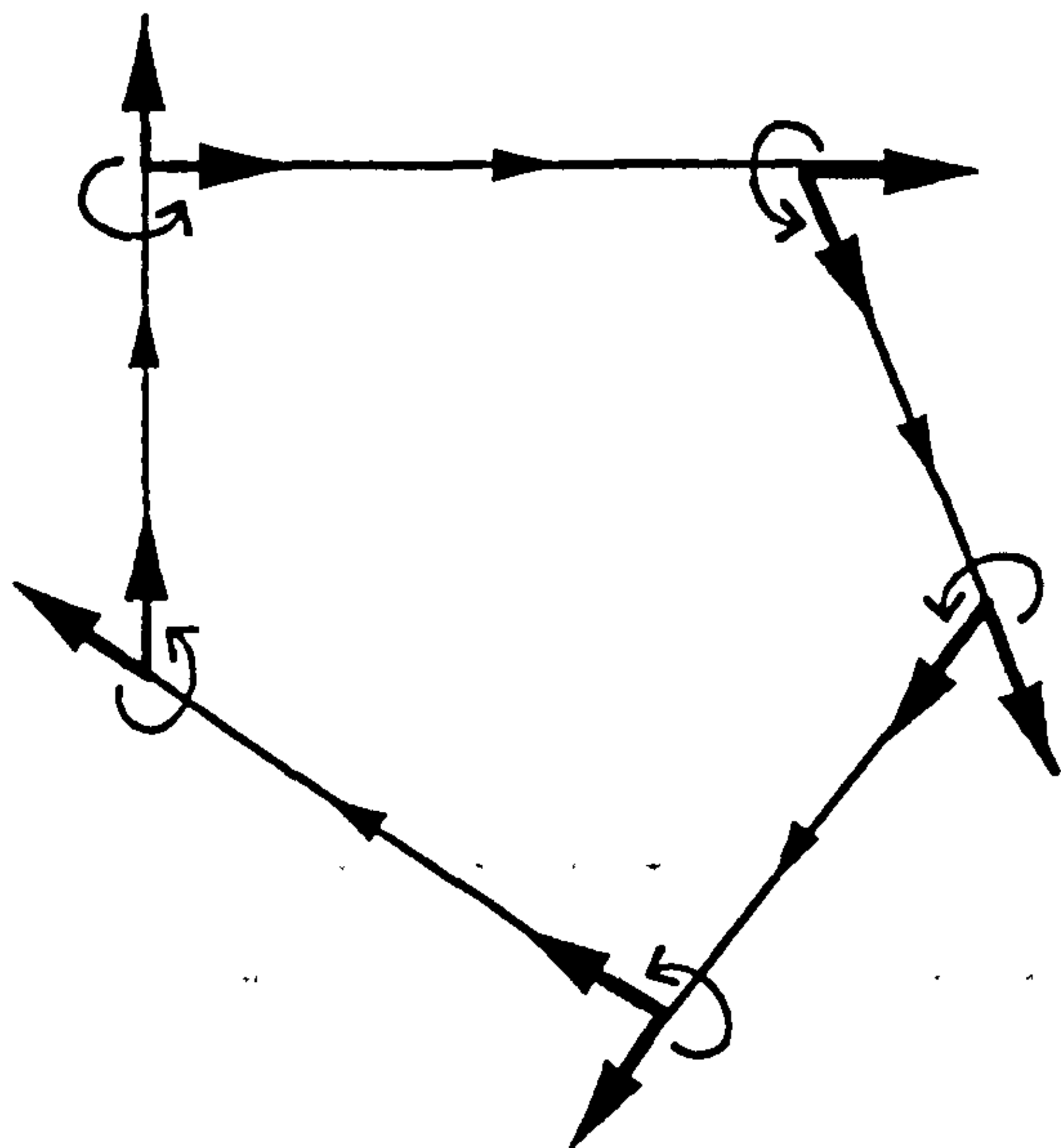
After 'make edge, kill loop':  
 Loop L1 is deleted, L2 becomes peripheral loop.

**Figure 8-13 Joining peripheral and hole loops**

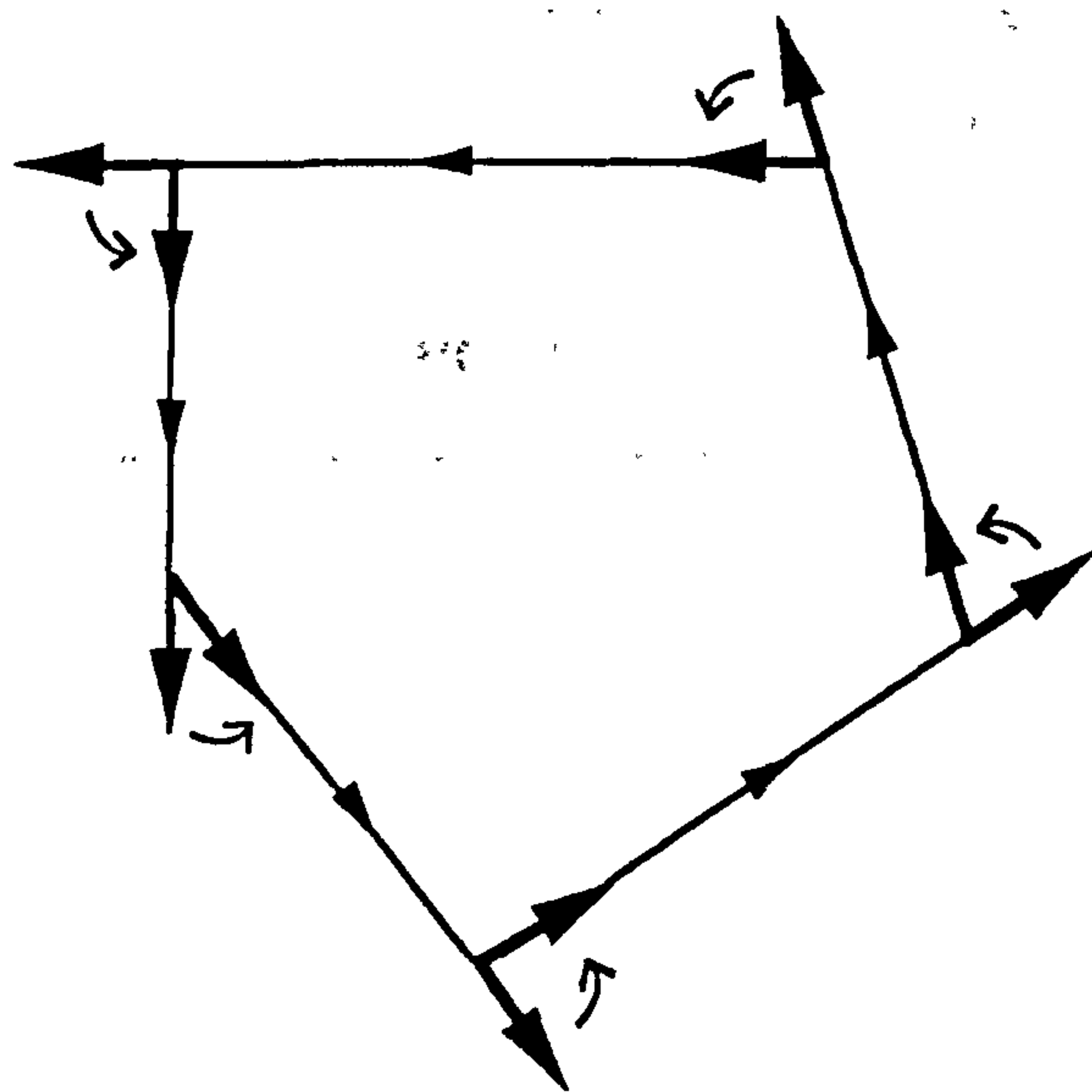


Resulting loop  $L$  is a hole loop.

**Figure 8-14** Joining two hole loops



Peripheral Loop - Edges are defined clockwise  
Sum of angles is  $-2\pi$



Hole Loop - Edges are defined counter-clockwise.  
Sum of angles is  $+2\pi$ .

Figure 8-15 Calculation for peripheral and hole loops



b) Checking whether a point is inside or outside a loop

The sum of the angles between pairs of adjacent vectors is calculated, where a vector is formed from the given point to each of the vertices of the loop, illustrated in figure 8-16. The counter-clockwise angle between each pair of vectors is calculated in the range  $-\pi$  to  $\pi$ . The total of these angles is  $2\pi$  or  $-2\pi$ , if the point is within the loop. Otherwise, the total is zero, and the point is outside of the loop. Note that this calculation is a geometric check, and there is no need to differentiate between hole and peripheral loops.

This calculation is required for 8.3.1 a) above, where any point of the hole loop is used to check whether the hole loop is totally within the new peripheral loop.

This check is also used to establish the face in which the first endpoint of a line lies, as specified in 8.1. A search is made of all the current internal faces and it is checked that the point is:

- i) Within the peripheral loop of the face, and
- ii) Outside of all hole loops of the face. See figure 8-17.

A point is 'in' the external face if it is outside all of its hole loops.

#### 8.4 Tolerance of Graph

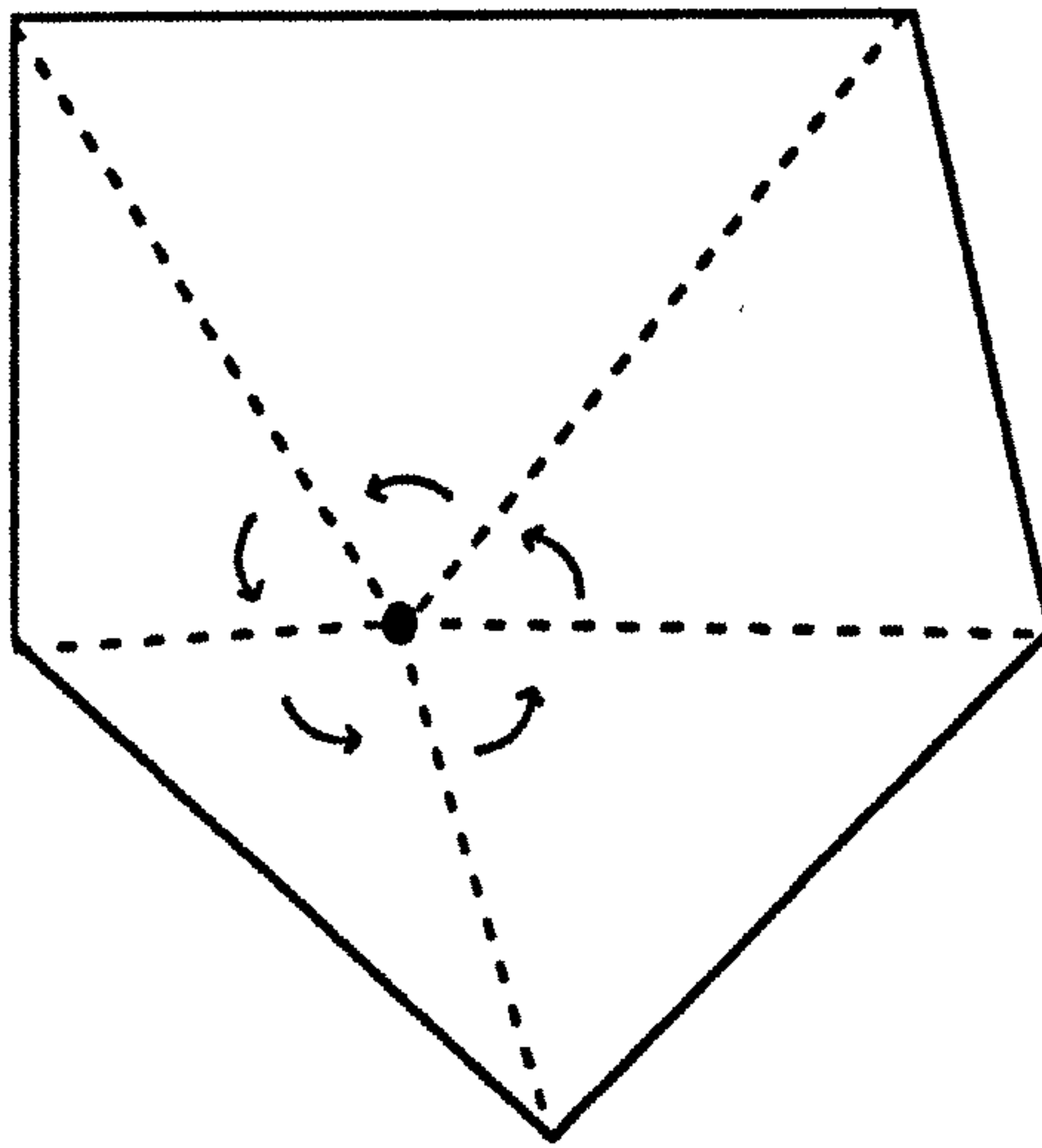
As stated in 8.1, it has to be established where the starting point of a new line is relative to the graph. To be able to determine this, it needs to be defined exactly what is meant by:

- 'a point is coincident with a vertex'
- 'a point lies on an edge'
- 'a point lies within a face'

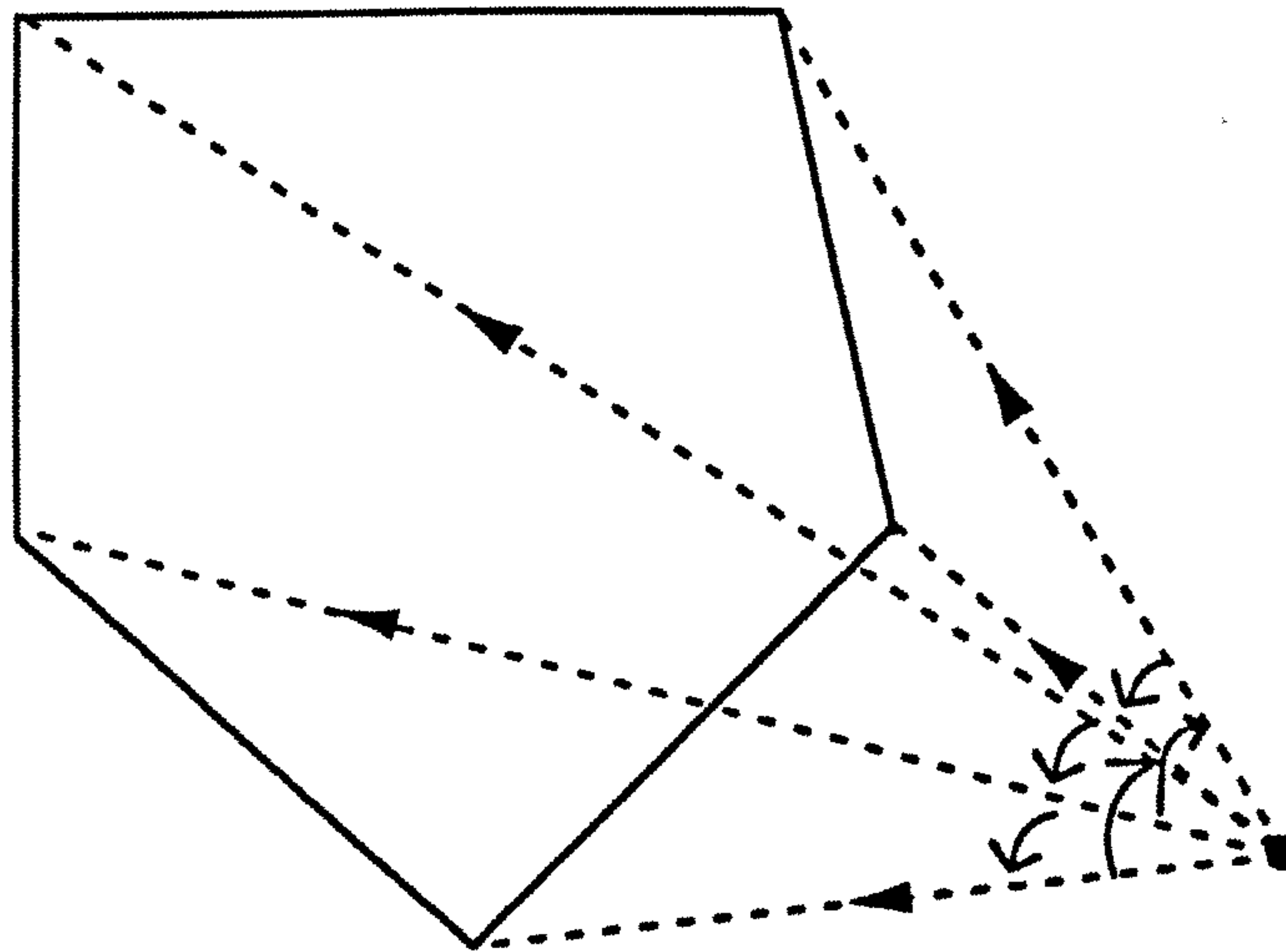
or expressed differently, how near does a point need to be to a vertex for it to be considered coincident, how near does a point need to be to an edge for it to be considered to be on the edge. The creation of the planar graph is an attempt to divide a continuous set of points on the plane into a discrete set of faces, edges and vertices.

Also to be considered is the fact that computers work within a limited accuracy and can produce rounding errors in calculations. It is necessary to be able to make calculations such as the normalised direction of an edge. If its two endpoints are too close together for the accuracy of the computer, either a 'divide by zero' error is possible in the calculation, or a vector is calculated which is incorrect, caused by dividing one very small number by another.

Therefore all geometric calculations need to be stable and consistent. A calculation cannot produce the result that a point is not on an edge, therefore is within a face, if the result of the calculation to check that the point is within the face is then deemed not true.

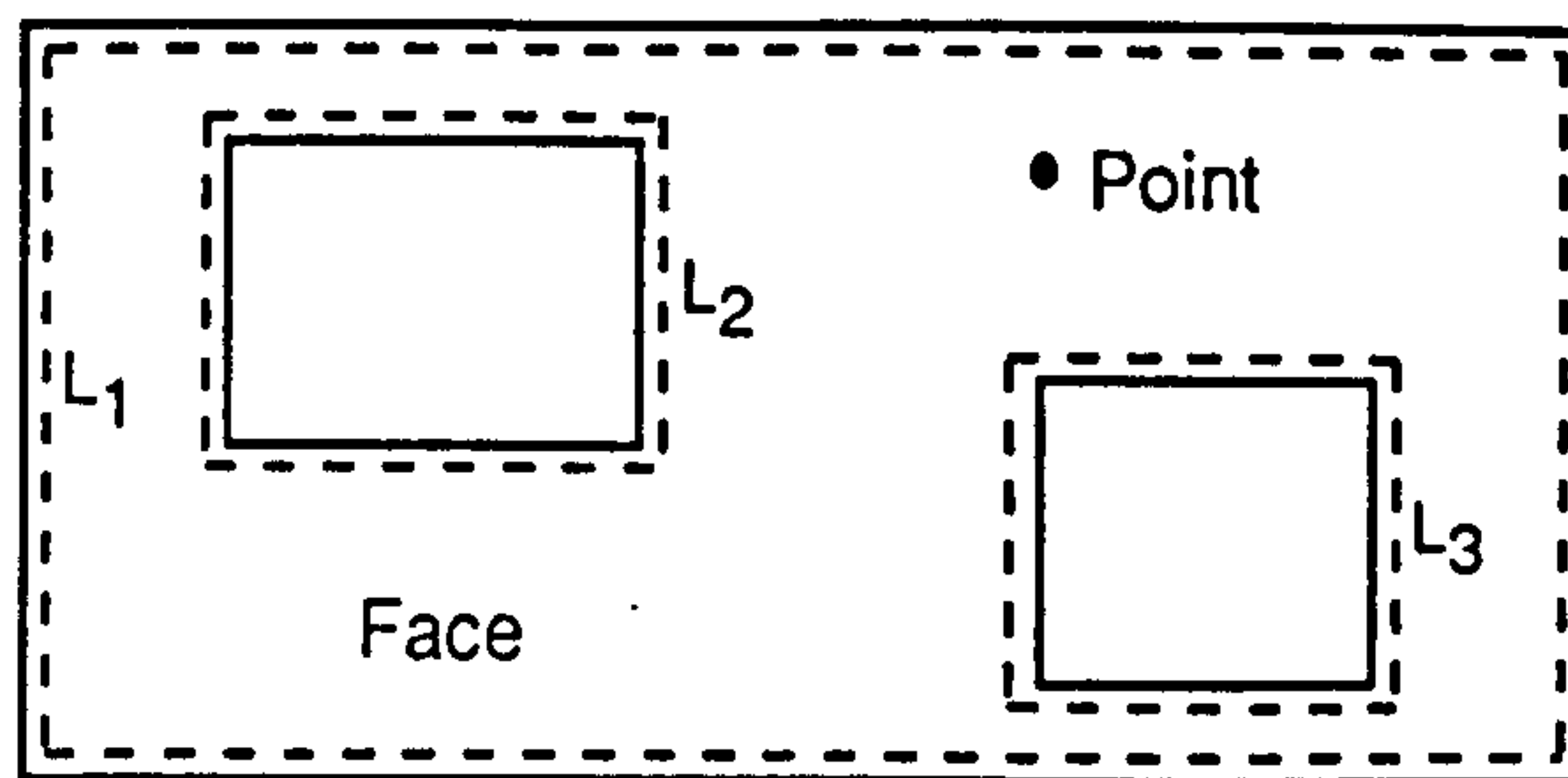


Point inside loop - sum of angles is  $+2\pi$  or  $-2\pi$



Point outside loop - sum of angles is 0.

Figure 8-16 Checking for point inside or outside of loop



Point is in face if ; 'inside' loop  $L_1$   
'outside' loops  $L_2$  and  $L_3$ .

**Figure 8-17** Checking a point is in a face

It becomes necessary to define a 'graph tolerance' to distinguish between points. Therefore the statement 'a point is coincident with a vertex', is interpreted as being true if the distance between the point and the point of the vertex is less than the 'graph tolerance'. A point is 'on an edge' if the point is coincident with a vertex of the edge or the perpendicular distance of a point to an edge is less than the graph tolerance. See figure 8-18.

#### 8.4.1 Finding Position of Point in Graph

Therefore, to establish the position of a point in relation to the vertices, edges and faces, the process is as follows:

a) Search through all vertices of the graph to find the nearest to the point. If the distance between the nearest vertex and the point is less than the tolerance, then the point is 'coincident with the vertex'.

b) Otherwise search through all the edges of the graph to find the nearest to the point, where 'nearest' depends on the perpendicular distance from the point to the edge. If the distance is less than the tolerance, the point is 'on the edge'. The point at which the edge would be split to create a new vertex is at the intersection of the perpendicular of the old point with the edge. This intersection point must be checked again for coincidence with the vertices of the edge. See figure 8-19.

Note that it is necessary to check all edges to find the nearest edge, as by the definition a point may be 'on' more than one edge. See figure 8-20.

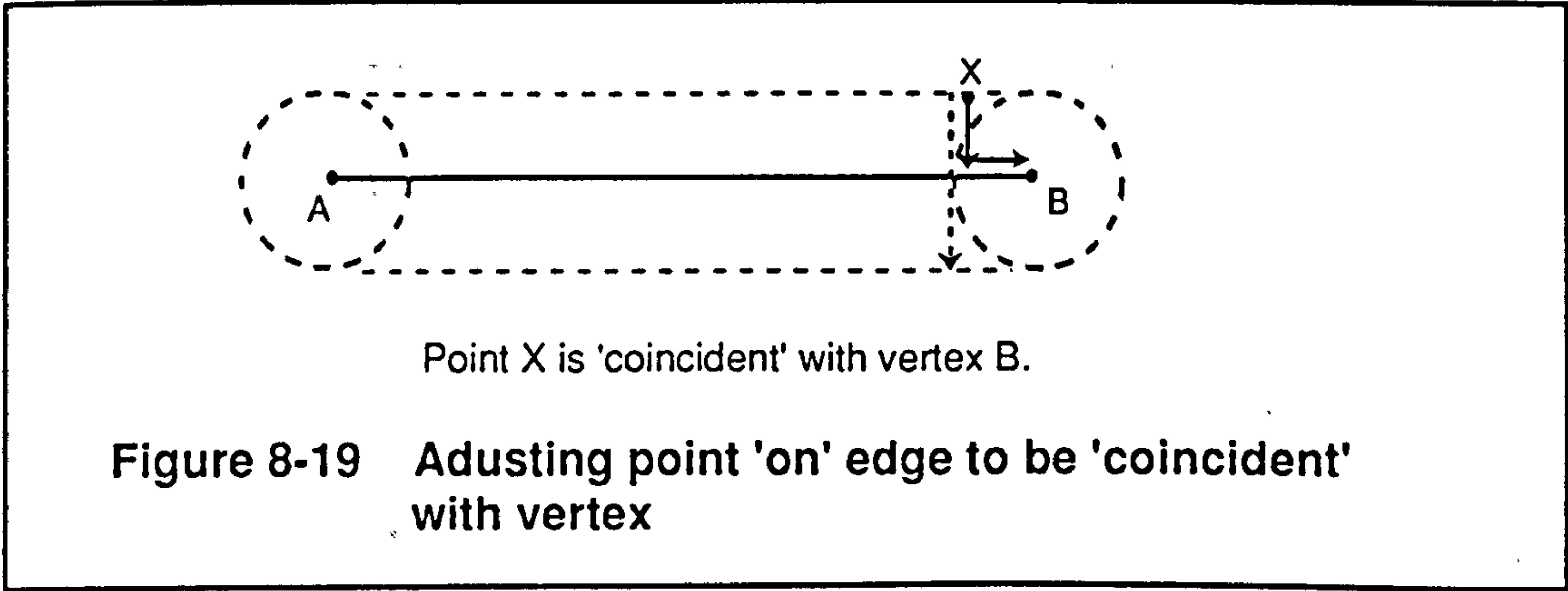
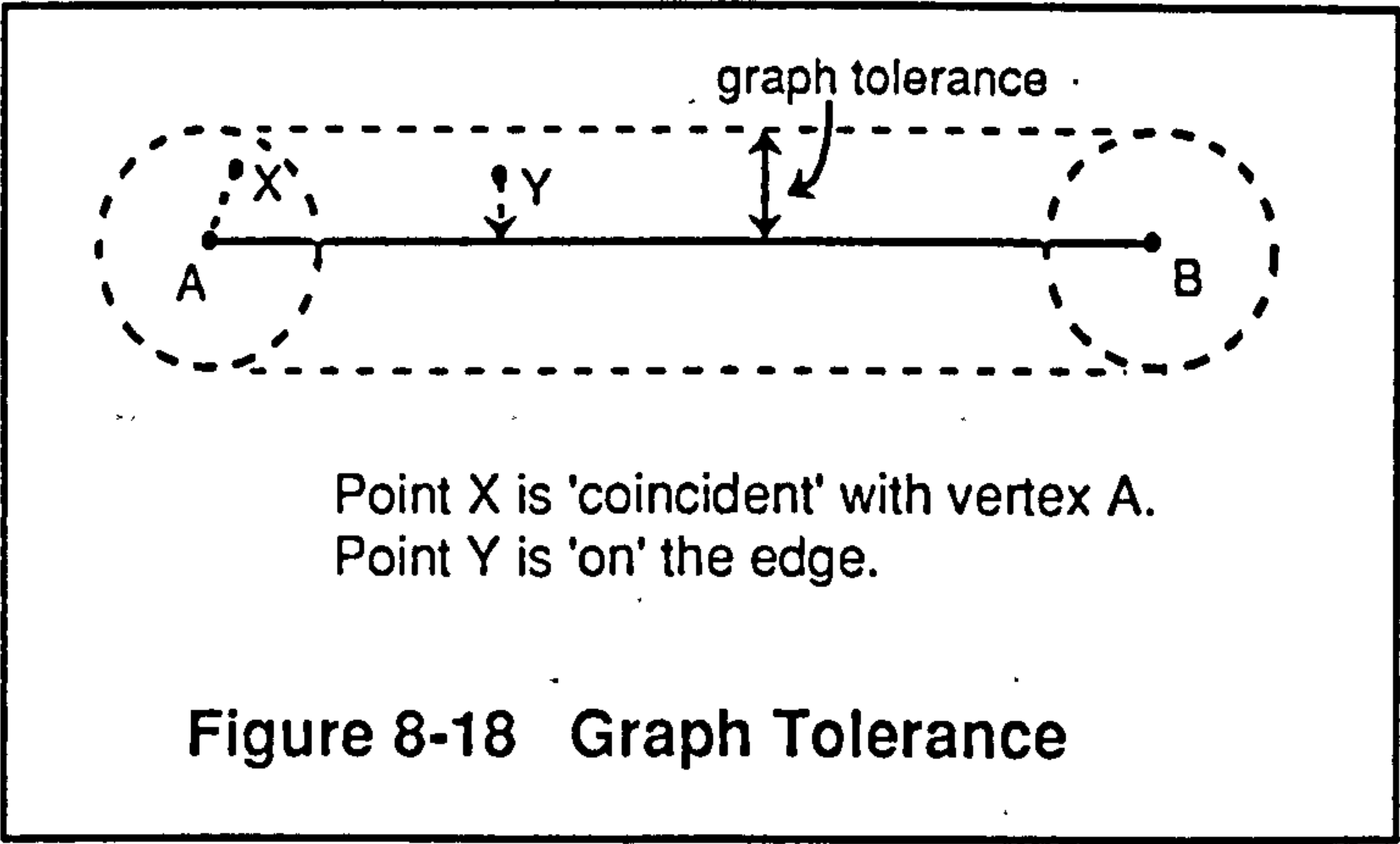
c) If a point is neither of the above, then using the technique described in 8.3.3, all faces of the graph are searched to find the face in which the point lies.

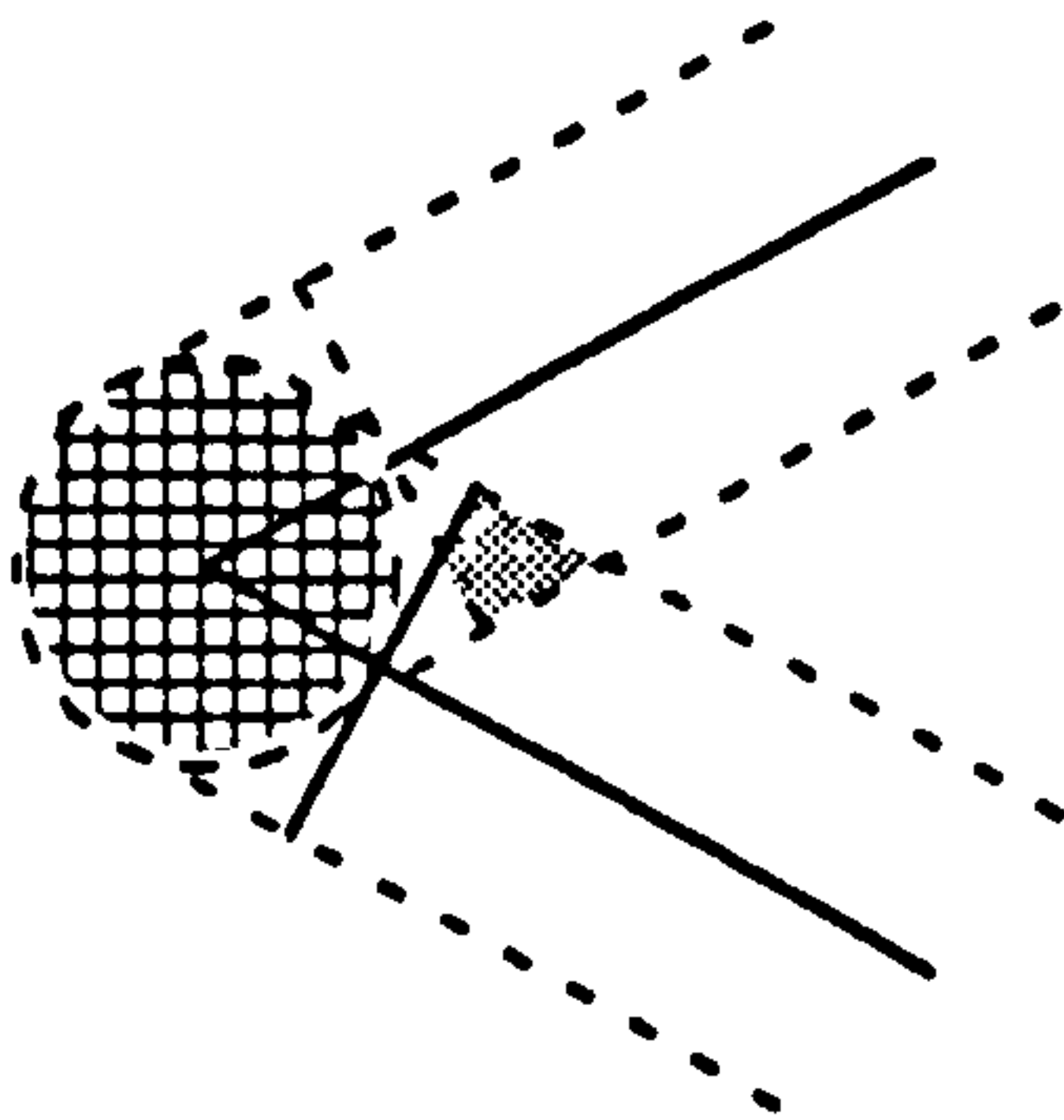
#### 8.4.2 Minimum edge length

From the above it is ensured that no edges are created less than the tolerance distance in length when splitting an existing edge, as edges are only split at points which are not 'co-incident' with one of its vertices, i.e. less than the tolerance from the vertex.

Therefore with an additional check that the distance from the 'starting vertex' of step 2, 8.1, to endpoint of line is greater than the tolerance for an edge to be created, this criteria is ensured for all edges.







Any point in hatched area is coincident with vertex.  
Any point in shaded area is 'on' both edges.

**Figure 8-20 Points on more than one edge**

### 8.4.3 Checking for Coincident Edges

A geometrical check is required to establish whether an edge is coincident with a line segment, when the line segment starts at a vertex of the edge. This is determined by one of the following being true:

- a) The endpoint of the line is 'on the edge'.
- b) The endpoint of the line is 'coincident' with the opposite vertex of the edge.
- c) The opposite vertex of the edge is 'on the line', where 'on the line' is defined as true if the perpendicular distance of a point to the line is less than the graph tolerance.

These cases are illustrated in figure 8-21.

### 8.4.4 Value of Graph Tolerance

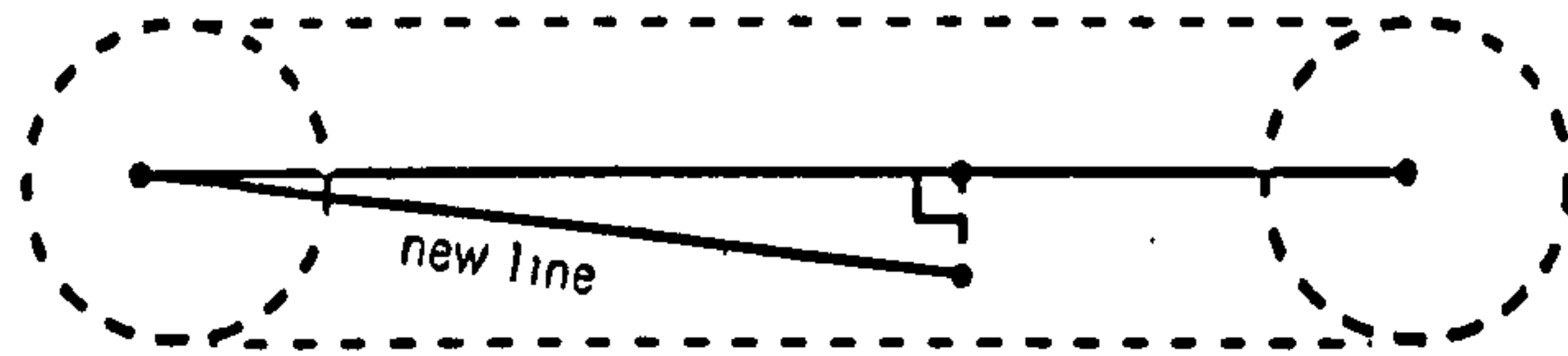
The graph tolerance value needs to be set before starting to create a graph and then held constant throughout its creation and modification to ensure consistency.

It must be at least large enough to allow for the accuracy of the calculations made by the computer, which is dependent on whether the implementation is in single or double precision. For single precision 6 decimal points is usually adequate.

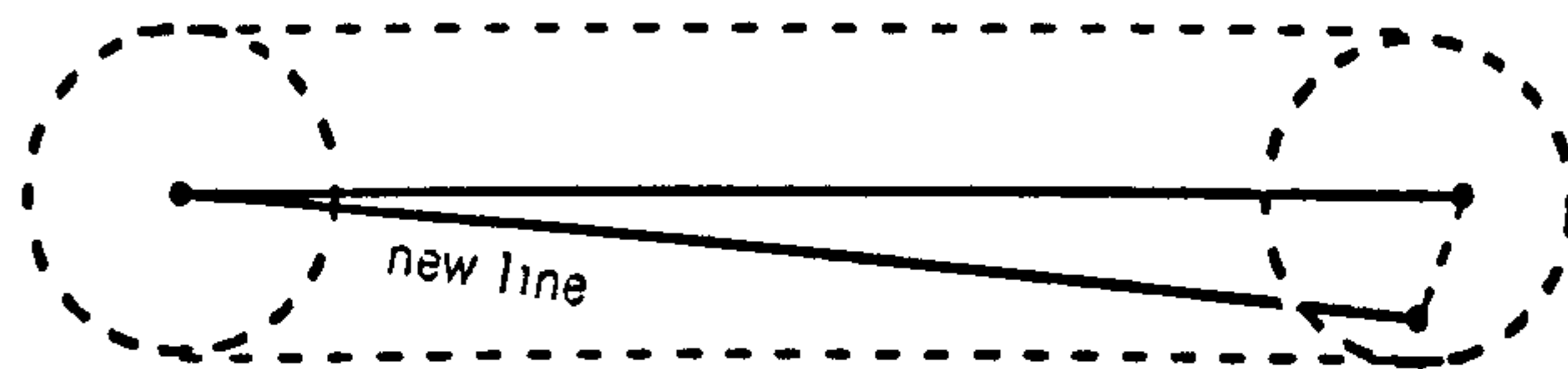
Its absolute value depends on the scale of the numbers being used to define the endpoints of the lines. If the assumption is that the floor of a building is being defined, where the points are in metres from an origin somewhere within the building, then this gives an indication that the magnitude of each co-ordinate is likely to be in the range 0 to 1000, assuming a building is likely to be less than 1 km in dimension. Therefore, a tolerance of at least 0.001 is most probably required.

Considering the fact that no edges will be created less than the tolerance in length, and that each edge represents a wall, it is unlikely that walls of length 0.001 metres would be desirable. A decision could be made that there would be no need to create walls less than say 0.1 metres. The wall thicknesses could also be taken into consideration, as when determining whether a point is on an edge, it is deemed so if the point is within the tolerance distance of the edge. This could be expressed as a point is on an edge when it is within the thickness of the wall represented by the edge. Therefore the minimum thickness of the walls can be used as a reasonable value for this tolerance, likely to be of magnitude 0.1 metres to 0.4 metres.

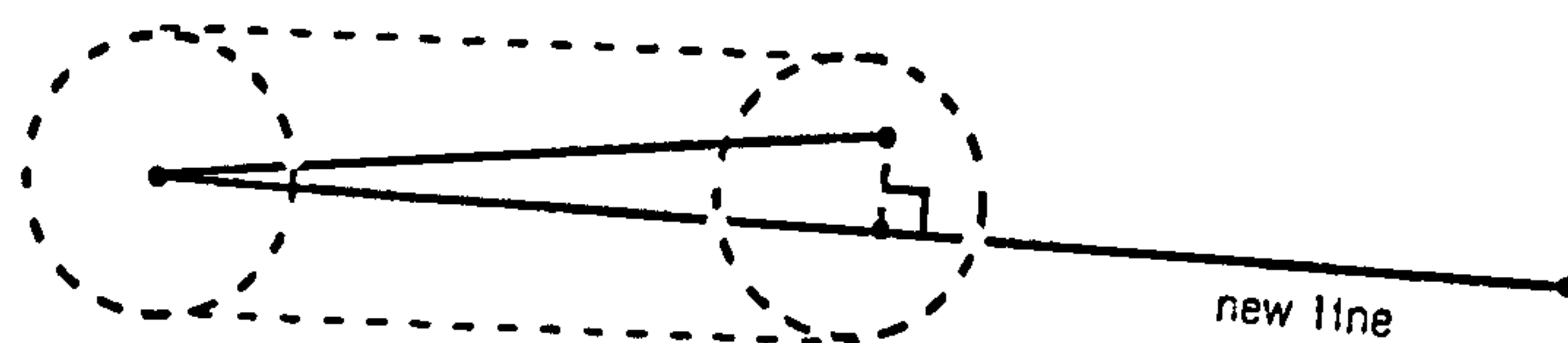
It appears that the tolerance value to be used in the creation of a graph is dependent on the model, and therefore should be varied for each graph. The best way to do this is to put it under the user's control, with a suggested default value of 0.2 metres.



a) Endpoint of line is 'on the edge'.



b) Endpoint of line is 'coincident with opposite vertex'.



c) Opposite vertex is 'on the line'.

**Figure 8-21 Checking for coincidence of edge with line**



## 9. CREATION OF BUILDING MODEL FROM PLANAR GRAPH

The process by which the spaces, nodes and constructions of a model are created from a planar graph representation of a floor is now described in detail. The creation of the standard entities of a winged-edge data structure is entirely through Euler operations, and these operations are described. The creation of the remaining data of the building model is also specified in detail: the space, node and construction data entities and the additional data fields of the winged-edge entities.

The following assumptions are made with regard to defining the process:

- a) A winged-edge data structure representing the planar graph of a floor has been created. This is a standard winged-edge data structure, such that the edges represent wall constructions, faces represent spaces and vertices represent vertical multi-edges. The geometry is defined by the 2D co-ordinates of the vertices, i.e. there is a geometry pointer in the vertex record to a 2D point entity.
- b) There are no wire edges, i.e. edges with the same face on either side. Although it would have been necessary for wire edges to exist during the creation of the graph, the assumption is made that the final result of the input of all the 2D line segments is that each face is defined by one or more loops of edges with no wire edges, such that a 2D profile can be created. The existence of wire edges indicate walls that are not forming part of the boundary of the space, and from the thermal analysis viewpoint, are only important in that they contribute to the thermal mass of the building, but no heat transfer needs to be modelled.
- c) The third dimension of the model is specified by the z position of the floor and by the height of the sweep.
- d) The external face of the graph is known, as it is not necessary to sweep this face.

There are two main steps to the creation process:

1. Creation of the geometry, i.e. 3D points and the node instance data.
2. Sweep operation to create the winged-edge data structure of each space, referencing the nodes created in step 1, and creating also the wall constructions referencing the faces.

### 9.1. Creation of Geometry and Nodes

Each graph vertex corresponds to a multi-edge, i.e. two nodes in the building model: a lower node at the z position of the floor, and an upper node at the

swept height. Each node is represented by a list of node instances, where a node instance is required for each adjacent graph face. See figure 9-1.

Node instance data for the entire graph, referencing the geometrical data i.e. 3D points, are therefore created first. Each graph vertex is considered in turn. For each graph vertex the process is:

1. Where  $(x,y)$  are the co-ordinates of the graph vertex,  $z$  is the floor position, and  $h$  is the floor height, create two 3D point entities, one at  $(x,y,z)$  and one at  $(x,y,z+h)$ .
2. Get list of adjacent faces of vertex.
3. To create the lower node, start a new list of node instance records. For each face, other than the external face, create a node instance record, and add to list. The lower 3D point entity is referenced by each node instance. See figure 9-2.
4. The upper node is created in the same way, but references the upper 3D point.

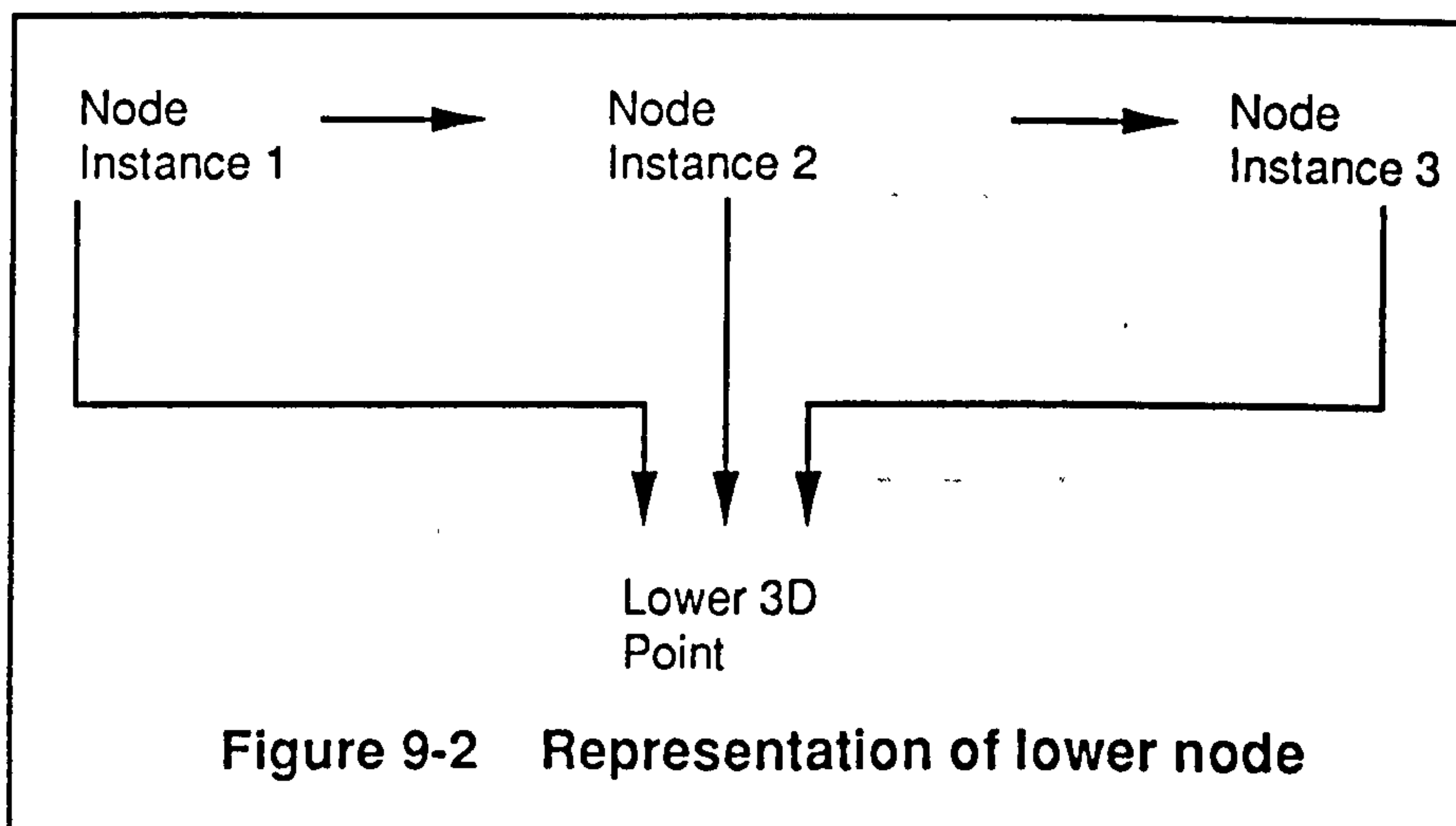
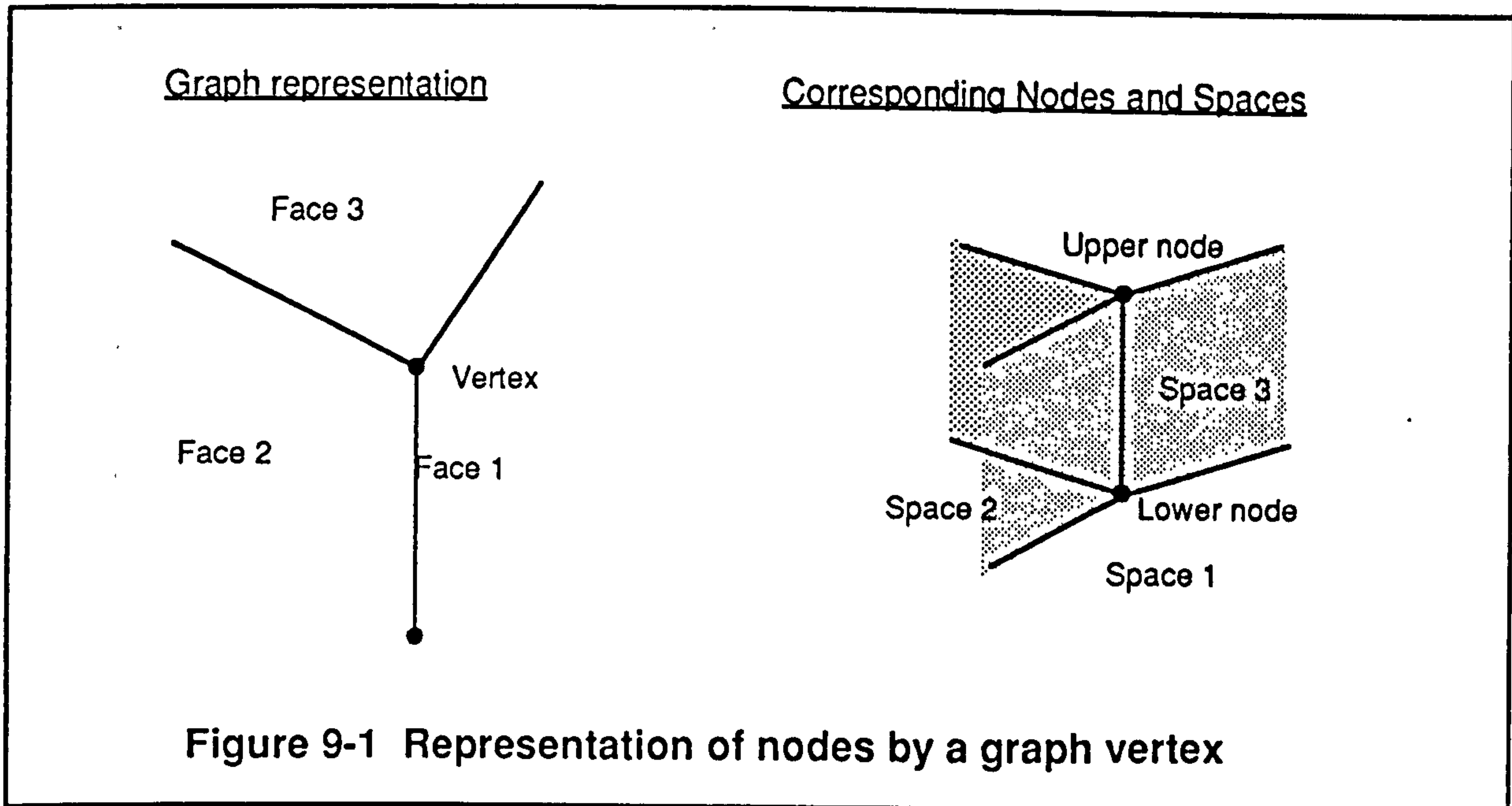
The node instances cannot yet reference a space vertex, as these have not yet been created.

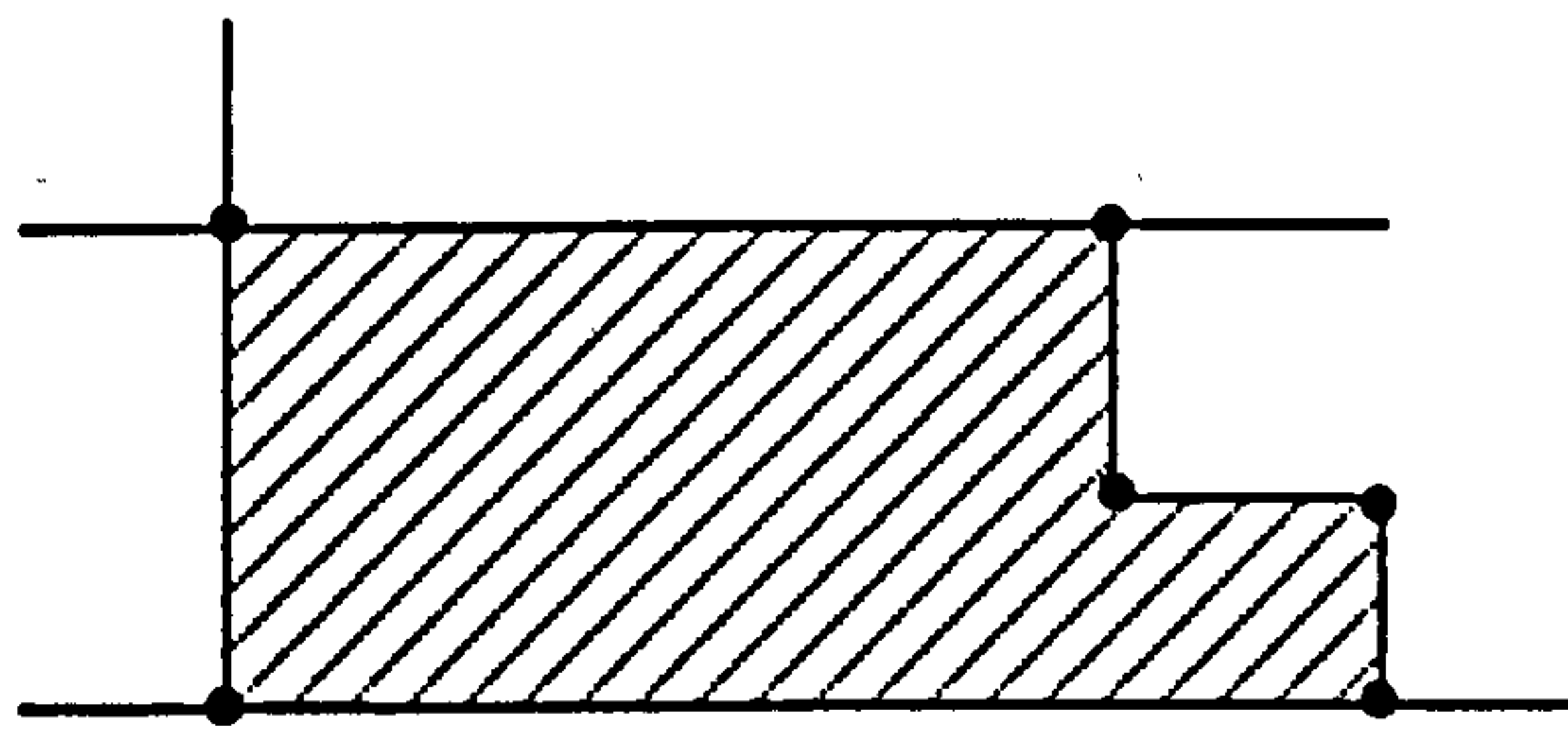
## 9.2 Creation of Spaces and Constructions

A space is created from each graph face, and a wall construction from each graph edge. Each graph face, except the external face, is considered in turn and the process for each face is now described. The Euler operations used to create the standard winged-edge entities are specified: these are in fact the same as those used in the normal profile sweep operation.

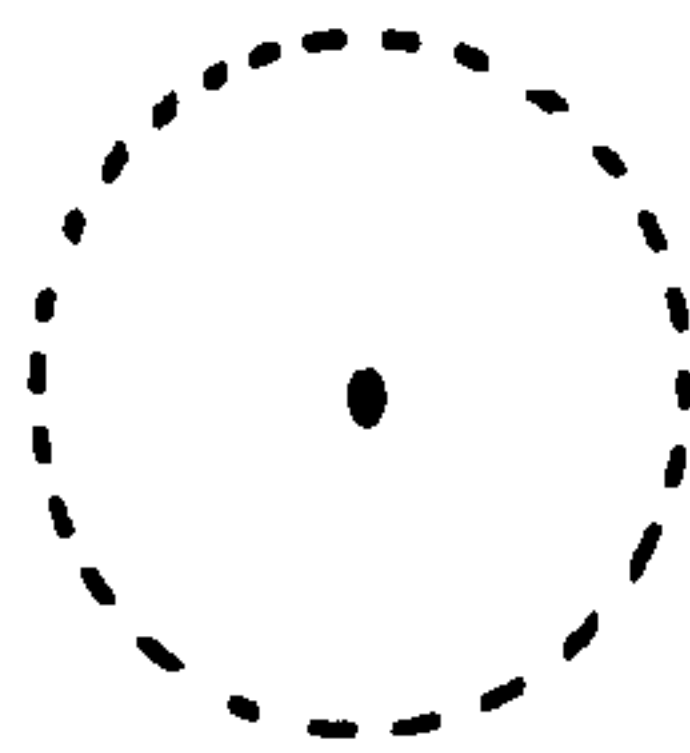
1. Create a new winged edge structure for a lamina of two faces, where the lower face of the lamina becomes the floor face of the space, and the upper face become the ceiling space (Figure 9-3):

- i) Euler operation 'Make vertex, face and body' creates initial topology, new face becomes floor face.
- ii) The edges of the peripheral loop of the graph face are copied, using 'Add an Edge and Vertex' operations, where each new edge is attached to the previous vertex. Note that no geometry is attached to the topology, the edges are only 'copied' in the sense that the same number are created to define a new loop and face.
- iii) The final edge is copied, using 'Add an Edge and Face' operation. The new face becomes the ceiling face.
- iv) Each hole loop of the graph face is then copied. The first vertex of each new loop is created with 'Add a vertex and create a hole loop', in the ceiling face.



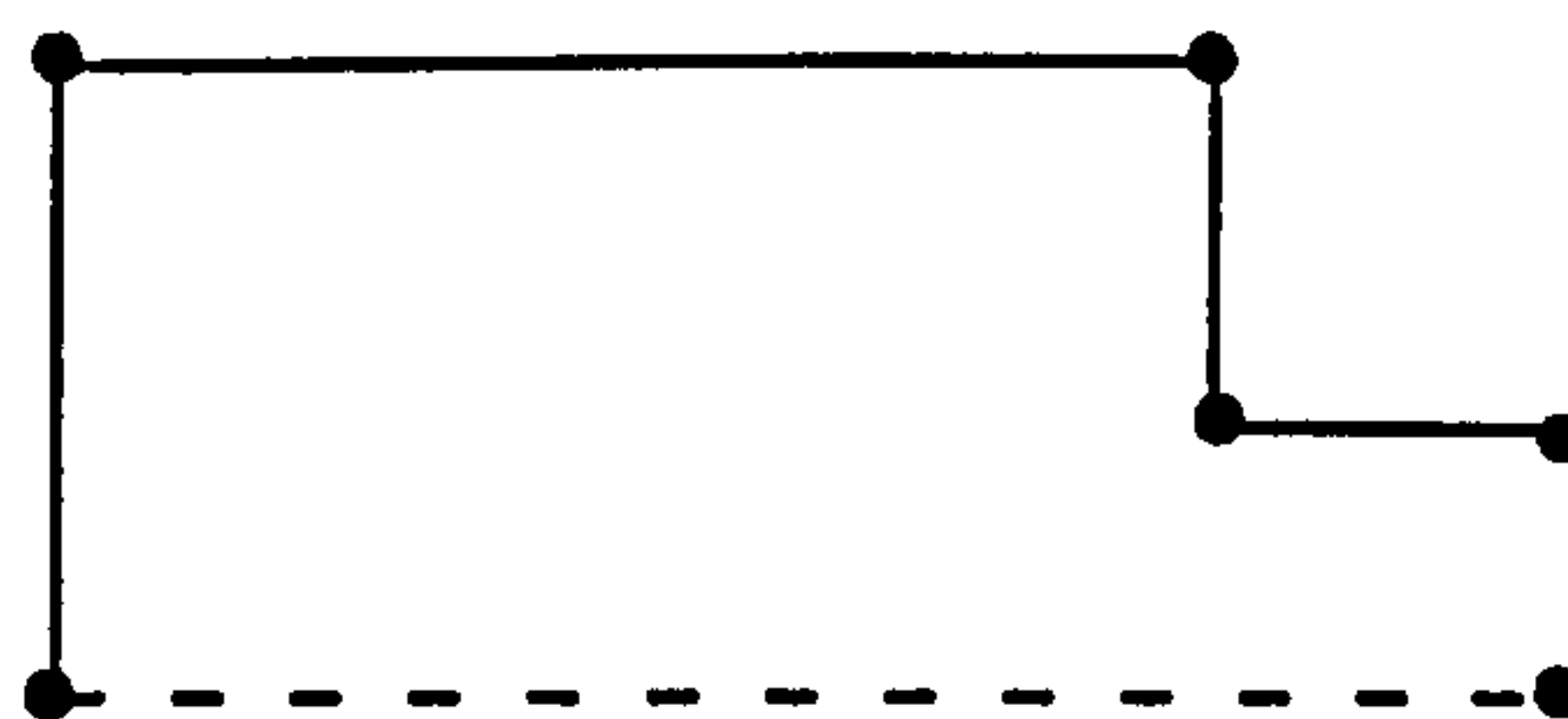


Graph Face



Floor Face

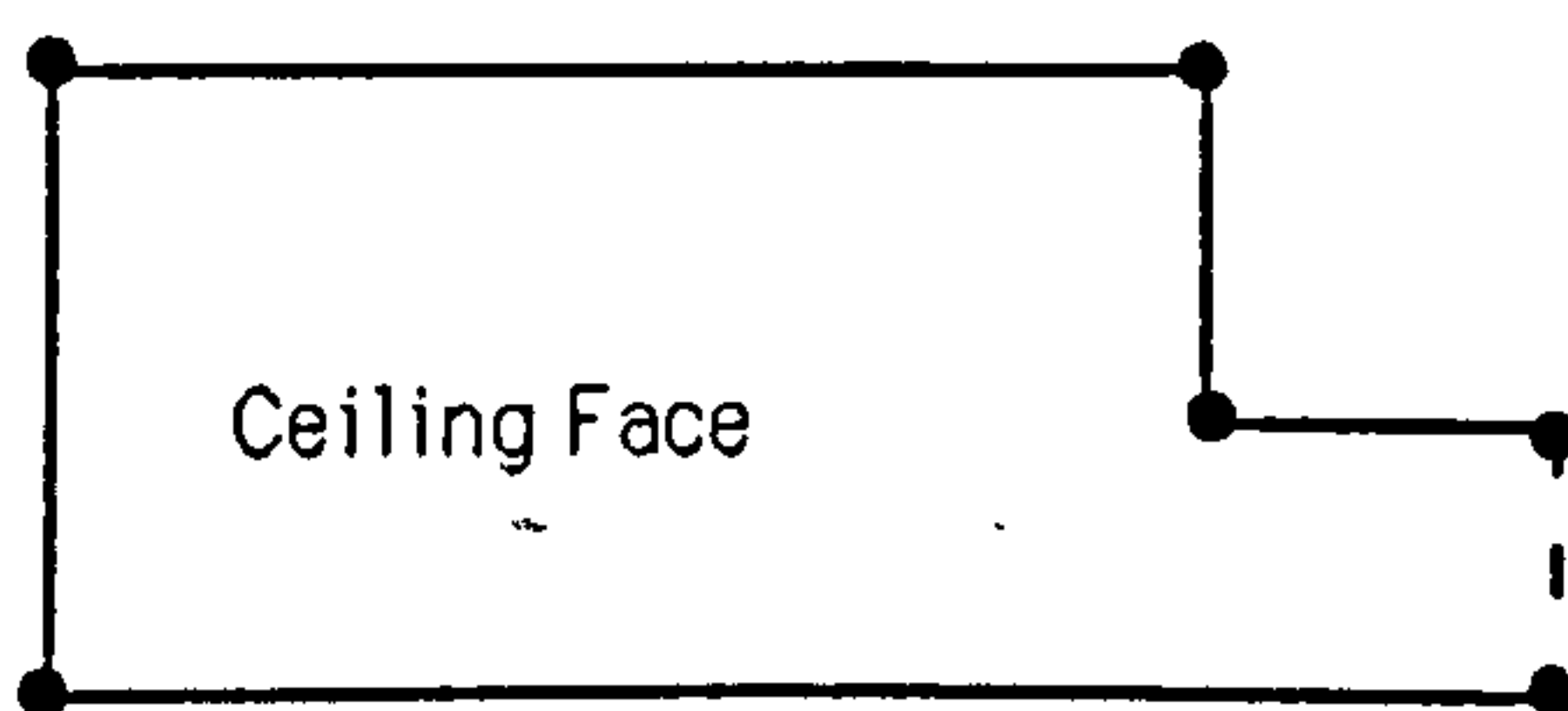
(i) Make vertex, face, body



New Edge

New Vertex

(ii) Add Edge and vertex



Ceiling Face

New Edge

(iii) Add Edge and Face

**Figure 9-3 Creation of 2 face lamina to represent new space**



v) The faces created in the previous step are then deleted by merging each one with the floor face: 'Make a hole loop, genus and delete a face'.

The resulting lamina consists of two faces, which are bounded by the same loop(s) of edges, where the number of loops and edges is equal to that of the original graph face.

## 2. Create space entity, referencing floor face (figure 9-4):

i) A new space entity is created, and added into the linked list for the building. It is referenced by the building if it is the first space of the building.

ii) The first face of the space is set to the floor face.

iii) Both the floor and ceiling faces reference the space.

## 3. Sweep ceiling face to create wall faces and constructions and to create references between node instances and vertices. For each loop of the face:

i) Get edges and vertices of loop and create references between each vertex and its corresponding lower node instance. See figure 9-5.

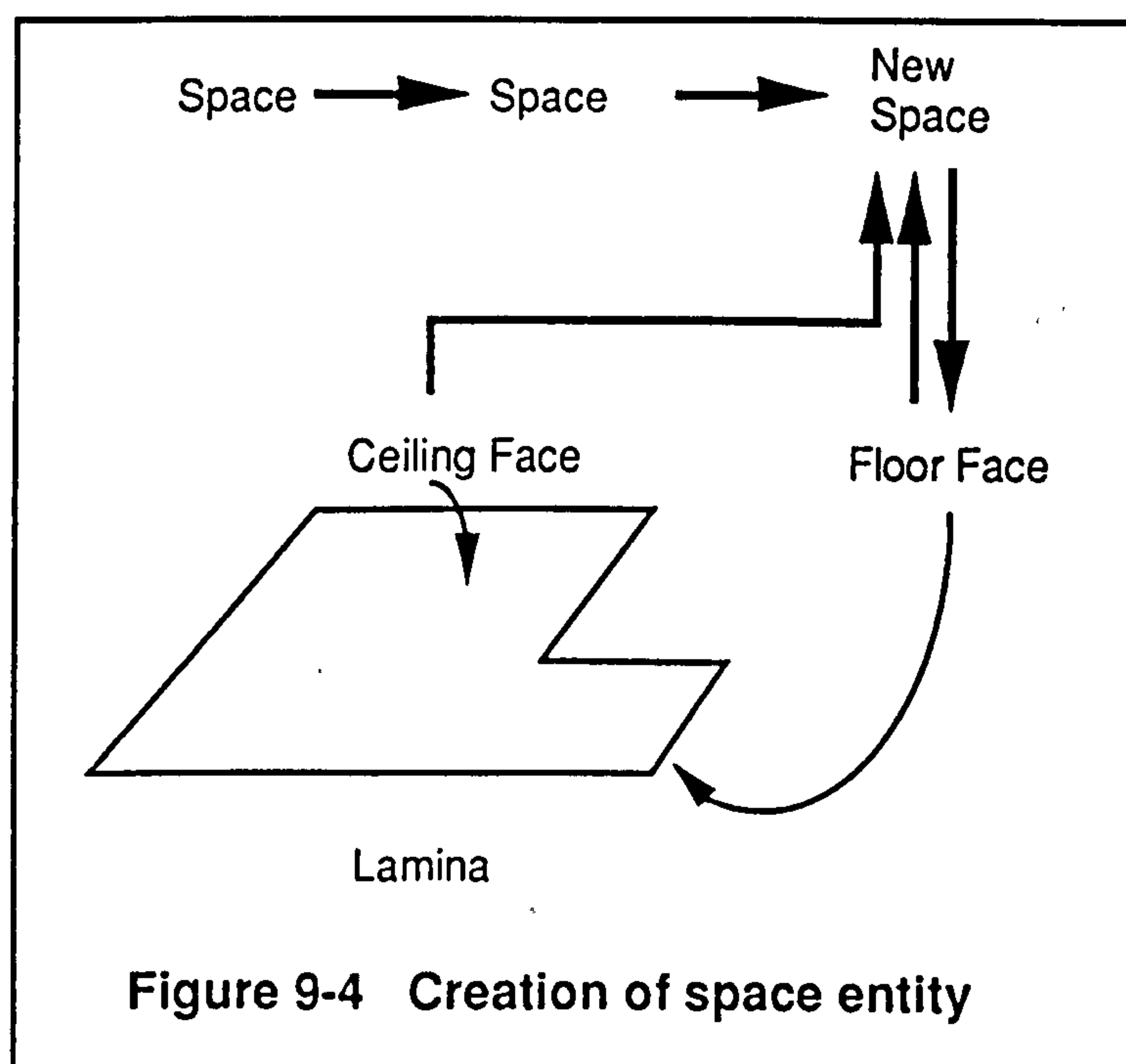
ii) For each edge of loop, create swept side face as in a normal sweep operation, using 'Add edge and vertex' to create the 'up' edge and top vertex, and 'Add Edge and Face' operation to create side faces. See figure 9-6.

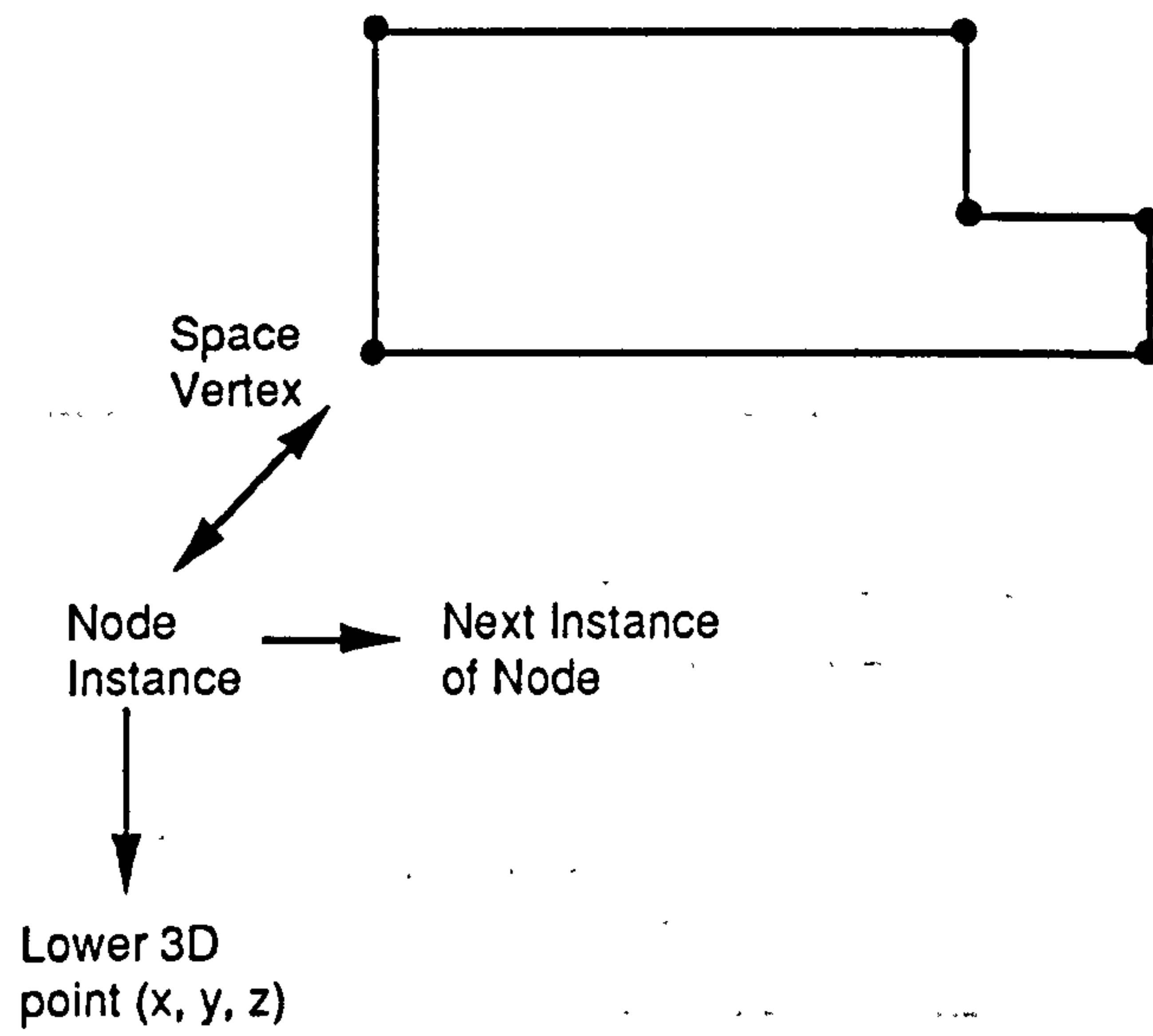
iii) Create references between each top vertex and the corresponding upper node instance.

iv) Create reference from each side face to new space entity. See figure 9-7.

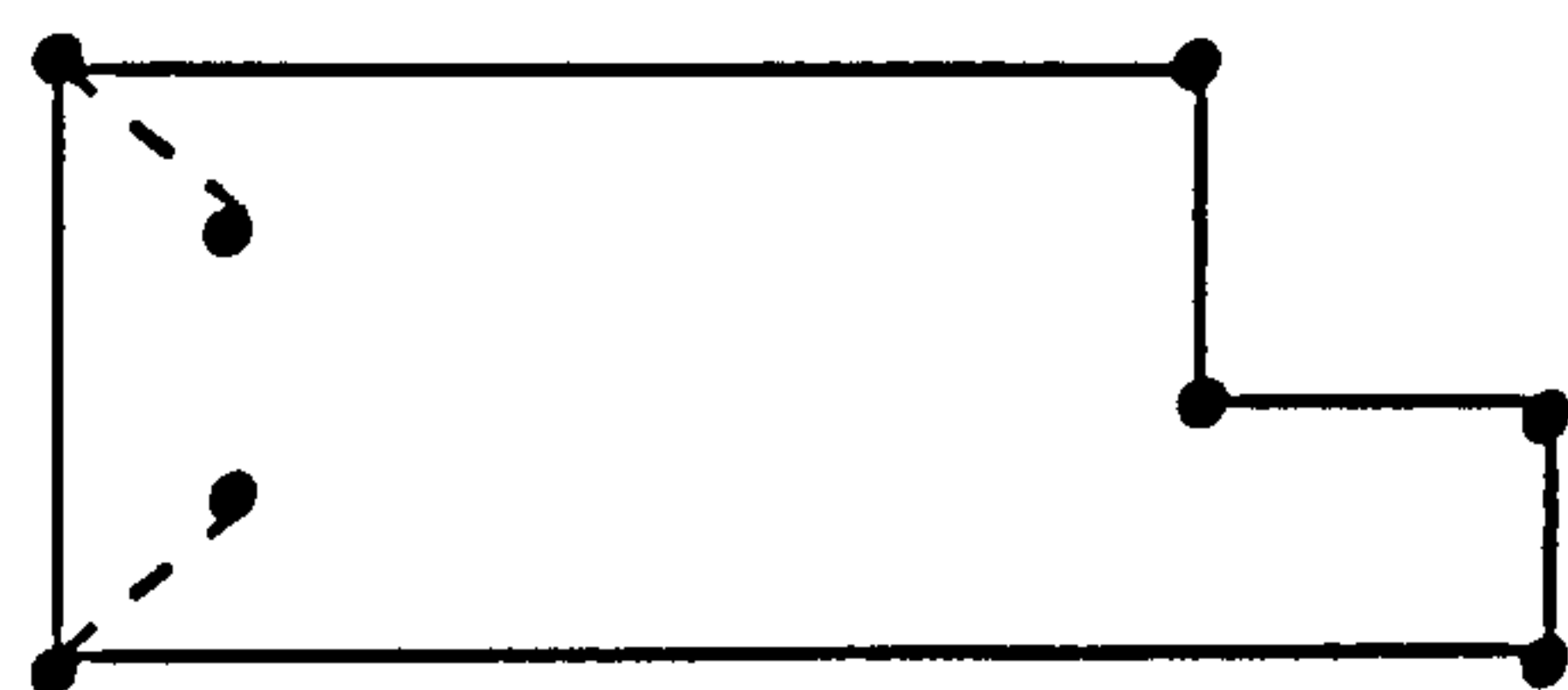
v) Create construction for side face if necessary. A construction needs to be created, if this is the first time that the original graph edge has generated a side face. That is the opposite graph face adjacent to the edge has not already been processed. Therefore, if required, a new construction entity is created, added into the list for the building, and references the new side face. The reference to the second face is initialised to zero. See figure 9-8.

vi) If the construction already exists, then the reference to the second face is set to the new side face.

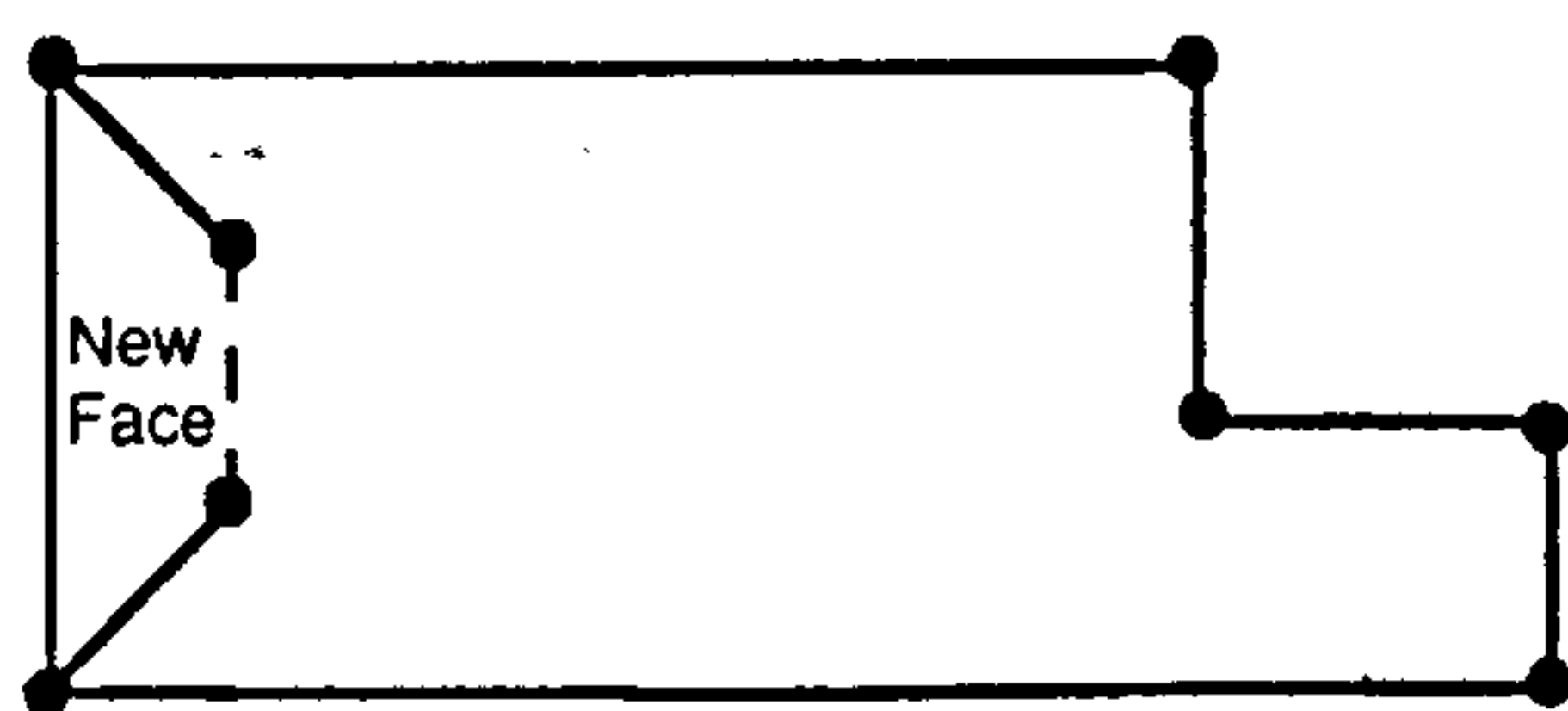




**Figure 9-5** Creation of references between space vertex and node

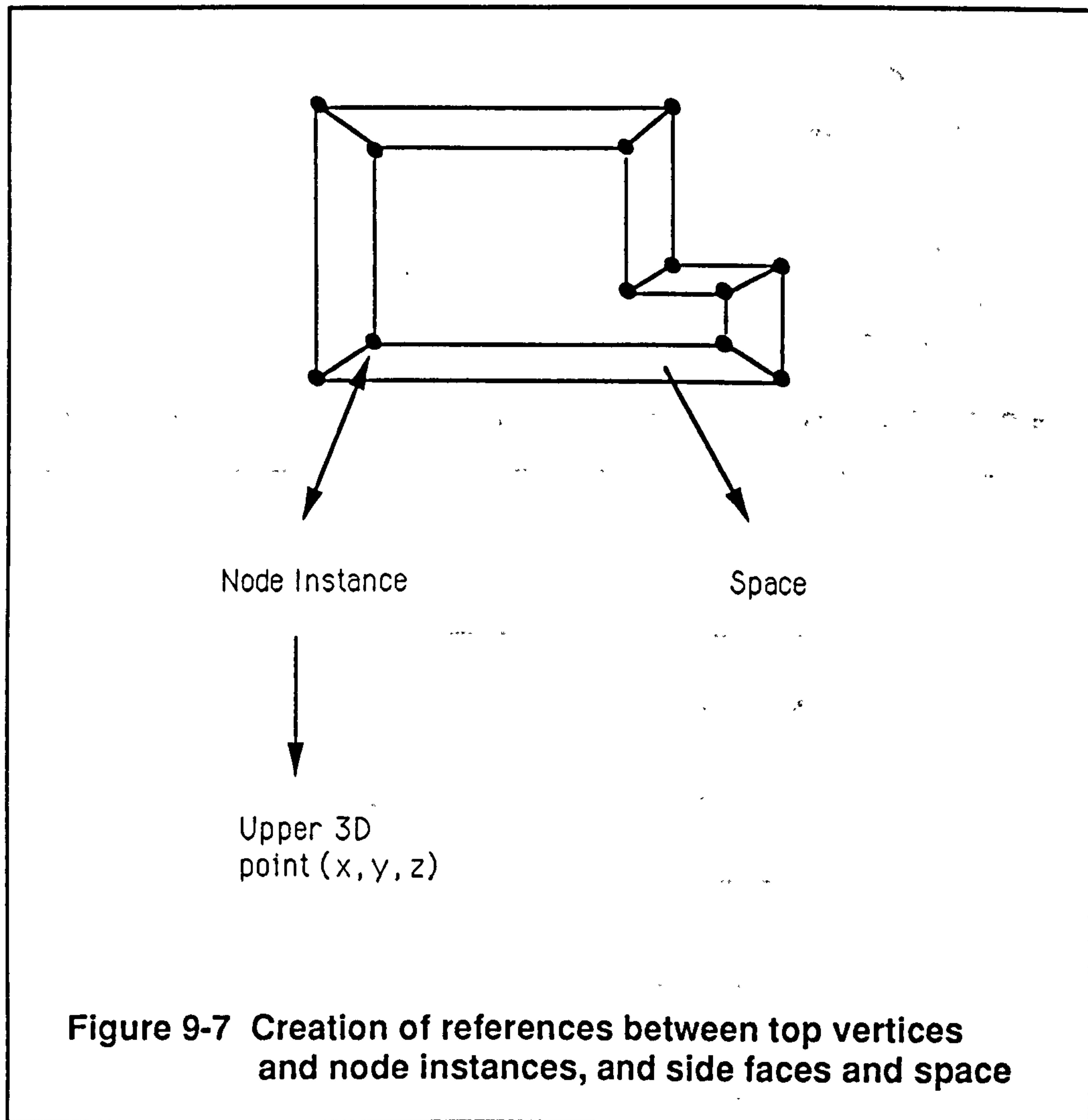


'Add edge and vertex' operations

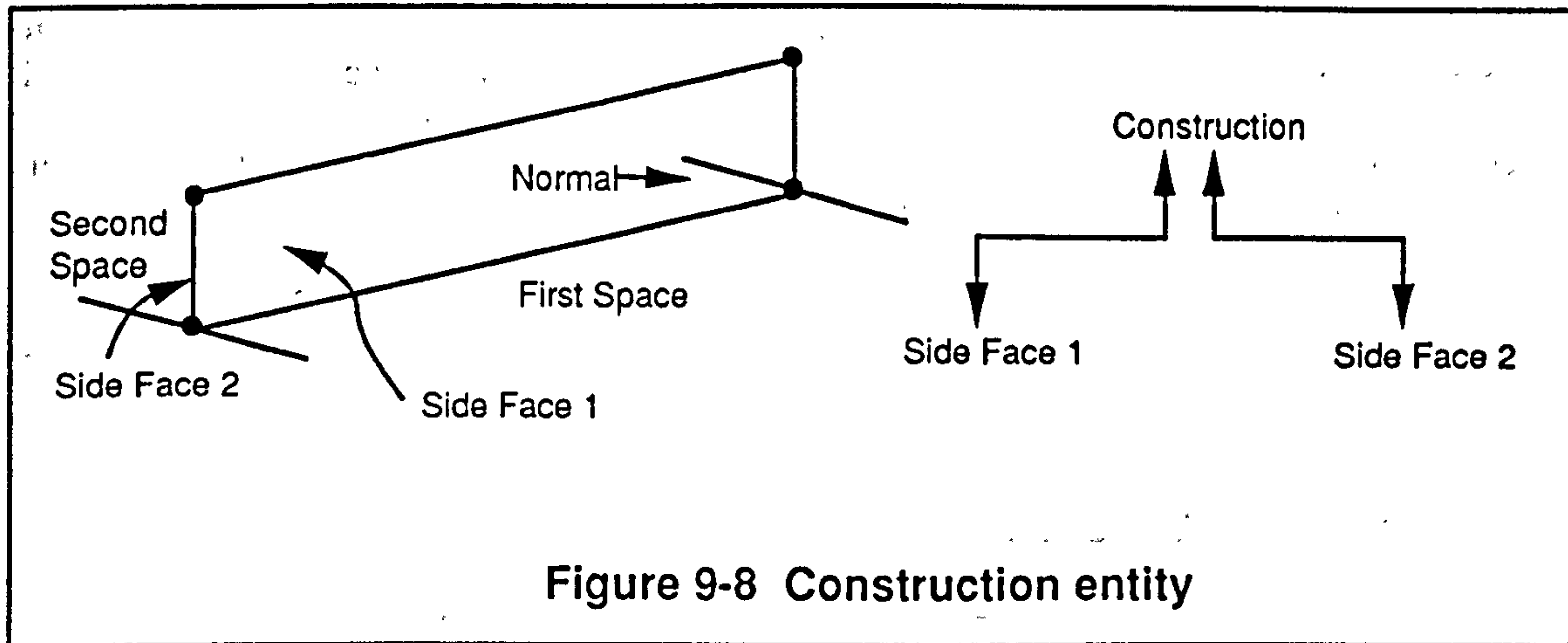


'Add edge and face' operation

**Figure 9-6** Creation of side face







vii) The normal of the construction is calculated from the 2D endpoints of the original graph edge, and is the direction perpendicular to that of the edge. The convention is adopted that the stored normal is that of the first referenced face pointing inside its space.

viii) The new side face references the construction.

To summarise, after the creation of one space as derived from a graph face, the space entity and winged-edge entities representing the space are completely defined. However the construction entities, referenced by the side faces, will not be complete until the spaces adjacent to the graph edges are created. When the process is complete for all graph faces, there will still be some constructions that only point to one face: these are the external constructions. The external graph face is not used and therefore a second side face for the construction is not created.

The node instance entities referenced by the space vertices are complete, but the node represented by a list of node entities will not be completely defined until all the spaces adjacent to the original graph vertices are created.

### 9.3 Temporary Data Pointers

As indicated above, construction and node data entities are in the state of being partially defined during the space creation process described above. In order to ensure their complete definition, it is necessary to retain temporary references between the new building data entities and the graph representation during the creation process.

#### 9.3.1 Graph Edges and Constructions

So that it can be determined whether a construction needs to be created at step 3v) above, the original graph edge has a temporary pointer to the construction derived from it. If this pointer does not exist, this indicates that this is the first time the graph edge has been processed, and a construction is created. The construction is then referenced by the graph edge, and used when the graph edge is processed for the second time.

#### 9.3.2 Graph vertices and Nodes

In order that references can be created between space vertices and node instances, as required in steps 3i) and iii) above, the pointers to the lower and upper node instances corresponding to one graph vertex and one of its adjacent graph faces are stored as temporary data. This is achieved by creating additional fields in the winged-edge representation of the floor graph, the requirement being to store two pointers, to an upper and lower node instance, for each vertex/face instance.

Therefore in each edge record, the node instance pointers for two vertex/face instances are stored:

- i) Face of right loop and next vertex instance
- ii) Face of left vertex and previous vertex instance. See figure 9-9.

The left face/next vertex and right face/previous vertex data are stored in the elcc and ercc edges of the edge shown.

Additionally each space edge created when forming the 2D lamina has a temporary pointer back to its originating graph edge, and an additional flag indicating whether it is in the same or opposite direction.

Therefore to locate the required graph face/vertex instance for a vertex of the lamina:

- i) The edge counter-clockwise from the vertex in the loop of the ceiling face references a graph edge.
- ii) Dependent on the flag associated with the edge, the required face/vertex instance is determined (Figure 9-10):

If the edge is the same direction as the original graph edge, the right face/next vertex instance data is accessed.

Otherwise the edge is in the opposite direction and the left face/previous vertex instance data is accessed.

#### 9.4 Use of winged-edge data structures

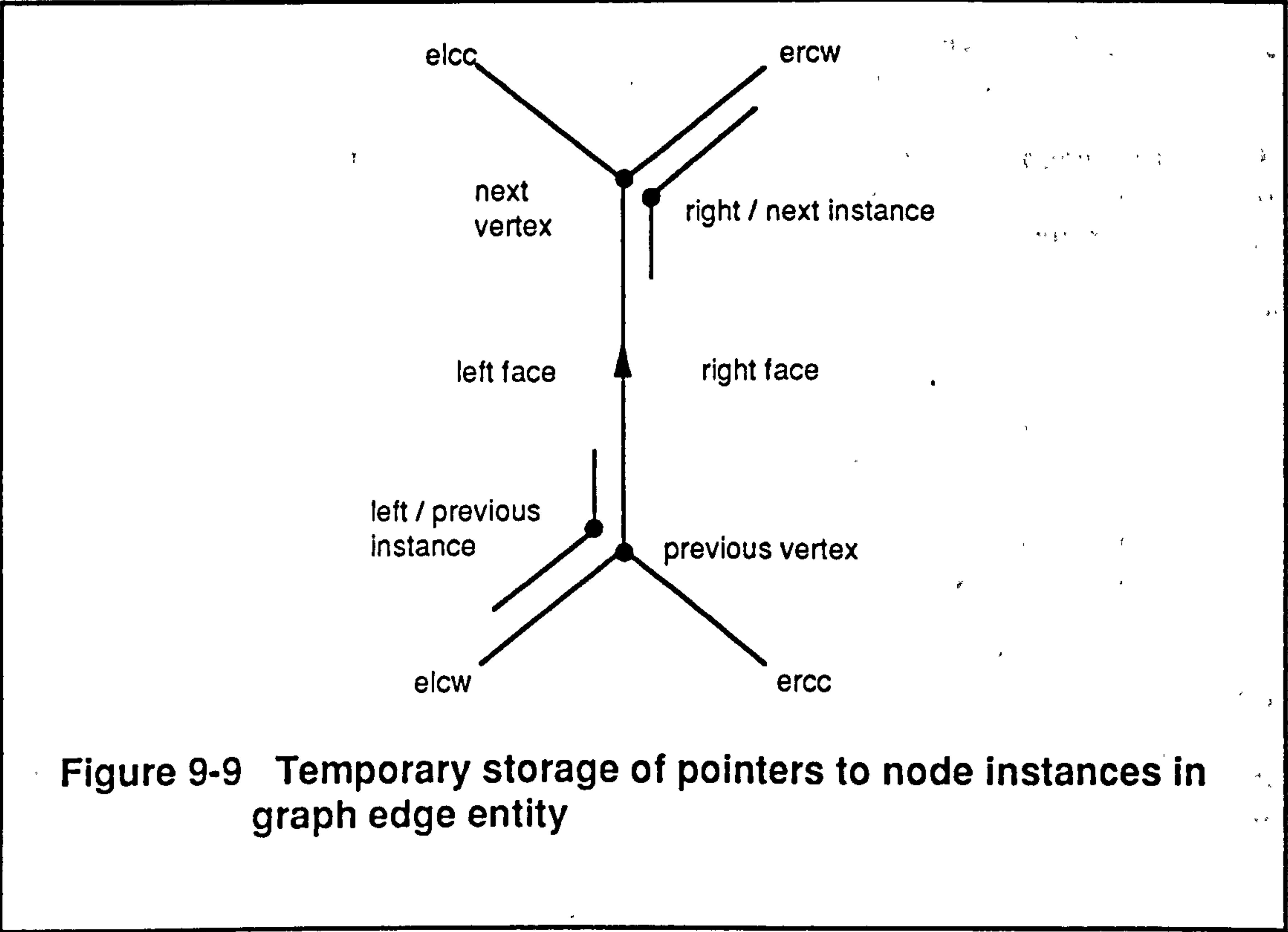
As described above and in Chapter 8, the winged-edge data structure is being used in two different ways:

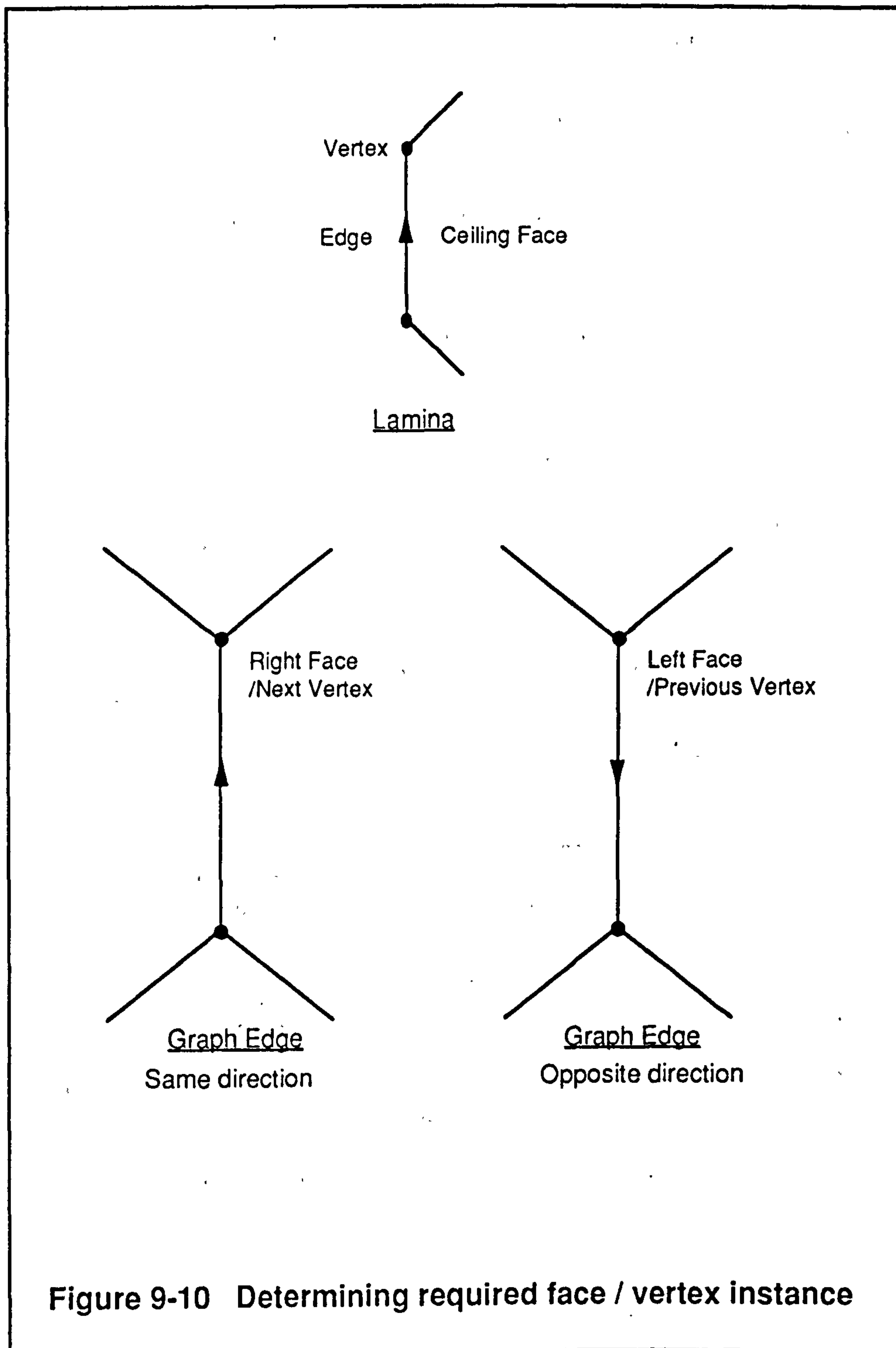
- i) to represent the enclosed boundary of a single space.
- ii) to represent the 2D planar graph of a floor.

Each usage requires additional fields to the 'standard' data fields of a winged-edge data structure as described in Chapter 5. The additional fields required for face, edge and vertex entities differ according to the usage. These are shown in table 9-1 the temporary data is indicated with \*.

	Space Representation	Floor Representation
Face	Space Construction	None
Edge	Graph Edge* Direction Flag*	Node Instances*
Vertex	Node Instance	2D point

Table 9-1: Usage of winged-edge data structure







Therefore, in order to keep the interface to the winged-edge entities well-defined, only through Euler Operations, these fields are grouped together to form new entities which are then referred to by a single pointer from the vertex, face and edge entities. This is similar to the 'geometry' pointer which is normally present, but indicates the 'usage' of the entity as well as its geometrical properties.

The following data entities are defined and include the fields above:

a) Space Face: pointers to space and construction forming the building model.

b) Space Edge: temporary data, only used during building model creation.

c) Graph Edge: only temporary data has been specified so far, although as will be described later, there is a further requirement to hold more data during the floor creation process by the user e.g. the allocation of construction types to wall constructions.

d) Graph Face: no data has been specified so far, although again there is a requirement to hold more data during the floor creation process to define the resulting space, e.g. zone number, floor and ceiling construction types.

To simplify vertices, the graph vertex can reference a node instance pointing to the required point. This instance is the only one in the list of the 'node'. There is therefore no need to create special space and graph vertex entities.

## 9.5 Specification of Functions performing Euler Operations

As described in section 5.9, the specification of the functions that perform the Euler Operations depends on the application environment from which they are called. Corresponding to the two usages of the winged-edge data structure, different sets of Euler Operations are also required. The required operations are indicated in table 9-2.

Euler Operation	Creation of Floor	Creation of Spaces
Make body, vertex, face mbvf	Yes	Yes
Add Edge and Vertex mev	Yes	Yes
Add Edge and Face mef	Yes	Yes
Add Edge, Delete Hole Loop mekh	Yes	No
Add vertex and Hole Loop mvh	Yes	Yes
Split an Edge (mev)	Yes	No
Make hole and genus, kill a face mhgkf	No	Yes
Kill edge and face kef	Yes	No
Make Hole Loop, kill an edge mhke	Yes	No
Merge edges (kev)	Yes	No
Kill vertex and hole loop kvh	Yes	No

Table 9-2: Euler Operations required for usages of winged edge data

The following points are therefore made, in consideration of the specification of the required Euler functions:

a) Instead of attaching geometry to newly created vertices, edges and faces, references to the 'usage' entities must be 'attached'. The required pointers can be generated before the Euler operation, and therefore each Euler function that creates a vertex, edge or face has these pointers as input parameters.

b) There are two operations that create an edge between two vertices: 'make an edge and face' and 'make an edge and kill a hole', although the latter operation is only used in the creation of the floor graph. As in this case the loops of the vertices being connected are known from searching all loops of a face for intersections (see 8.1, step 3), the required operation can be determined. Therefore these 'Add Edge' operations are implemented by two separate functions.

c) It is only in the creation of the floor graph that it is necessary to move hole loops between faces, when creating a new face. Therefore, a separate function is implemented which only moves loops between faces.

The required functions have therefore been specified, such that the same function for an operation is used in both the floor graph and building model creation.

## 10. FURTHER MODELLING REQUIREMENTS

The result of the process described in Chapter 9 is a model of a building with one floor and a horizontal roof. The model consists of spaces and wall constructions only. There are further geometric modelling requirements which must be met to enable the representation of a wide range of building types:

- a) Multi-floor buildings
- b) Complex 3 dimensional roof geometries e.g. pitch roofs
- c) Windows.

Requirements a) and b) are now discussed in more detail, c) is considered in conjunction with the allocation of construction attributes in the following chapter.

### 10.1 Multi-floor Buildings

Assuming that each floor of a building can be represented by a planar graph, and a model created of each floor by the process previously described, a complete building model is created by 'joining' these floor models into one building model. This is achieved by creating the floor/ceiling construction between the floors, where each construction refers to a ceiling face of a space of the lower floor model, and a floor face of a space of the upper floor model.

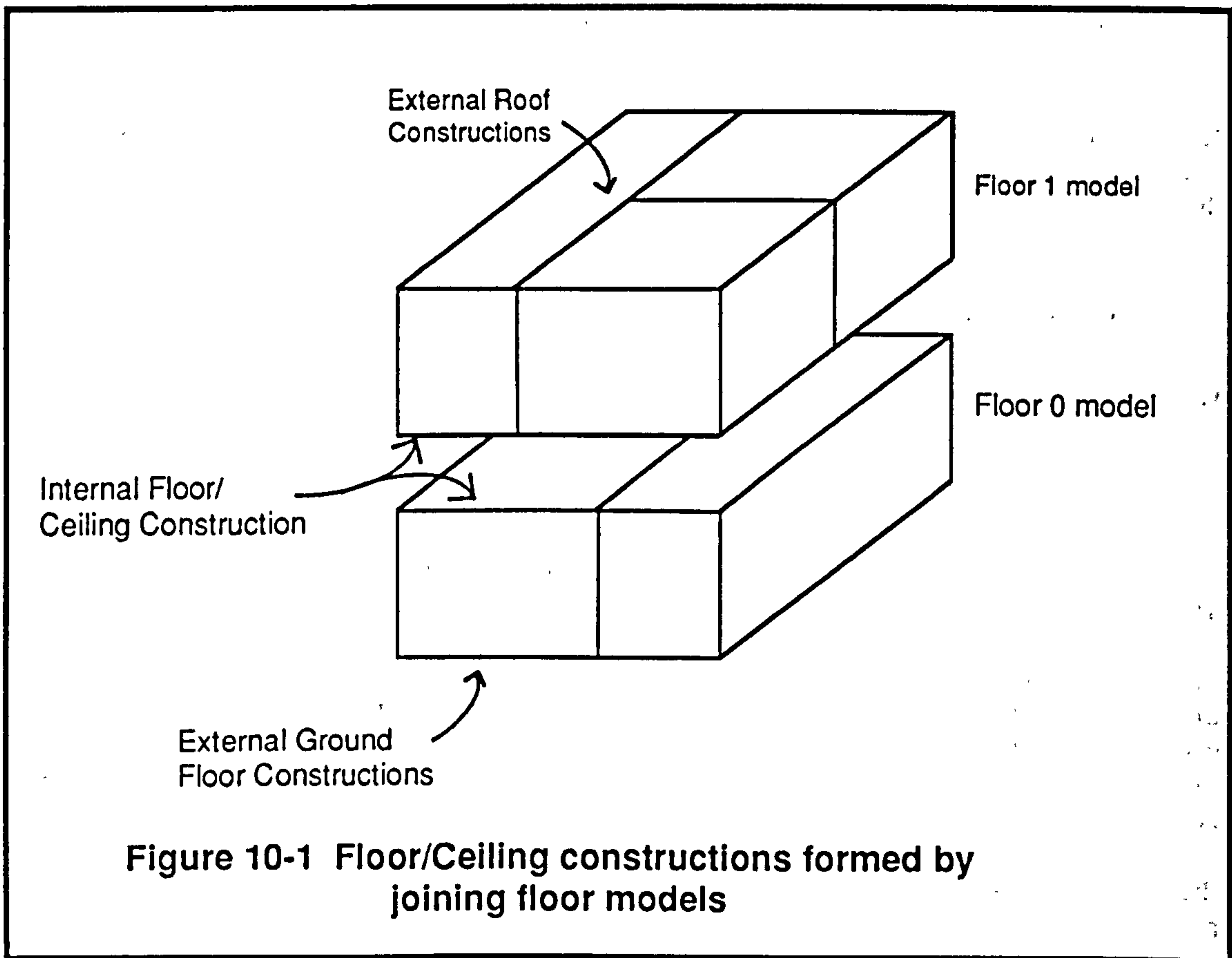
There will be some floor/ceiling constructions that only refer to one face, as there are similarly wall constructions. These form part of the external building envelope, for example, the floor constructions adjacent to the ground refer to a floor face only, and roof constructions refer to a ceiling face only. See figure 10-1.

The process to create the constructions requires a comparison of each ceiling face of the lower floor with each floor face of the upper floor, such that any overlap can be detected, and the faces divided appropriately. Figure 10-2 shows the resulting internal constructions created from the floor and ceiling faces of two floor models.

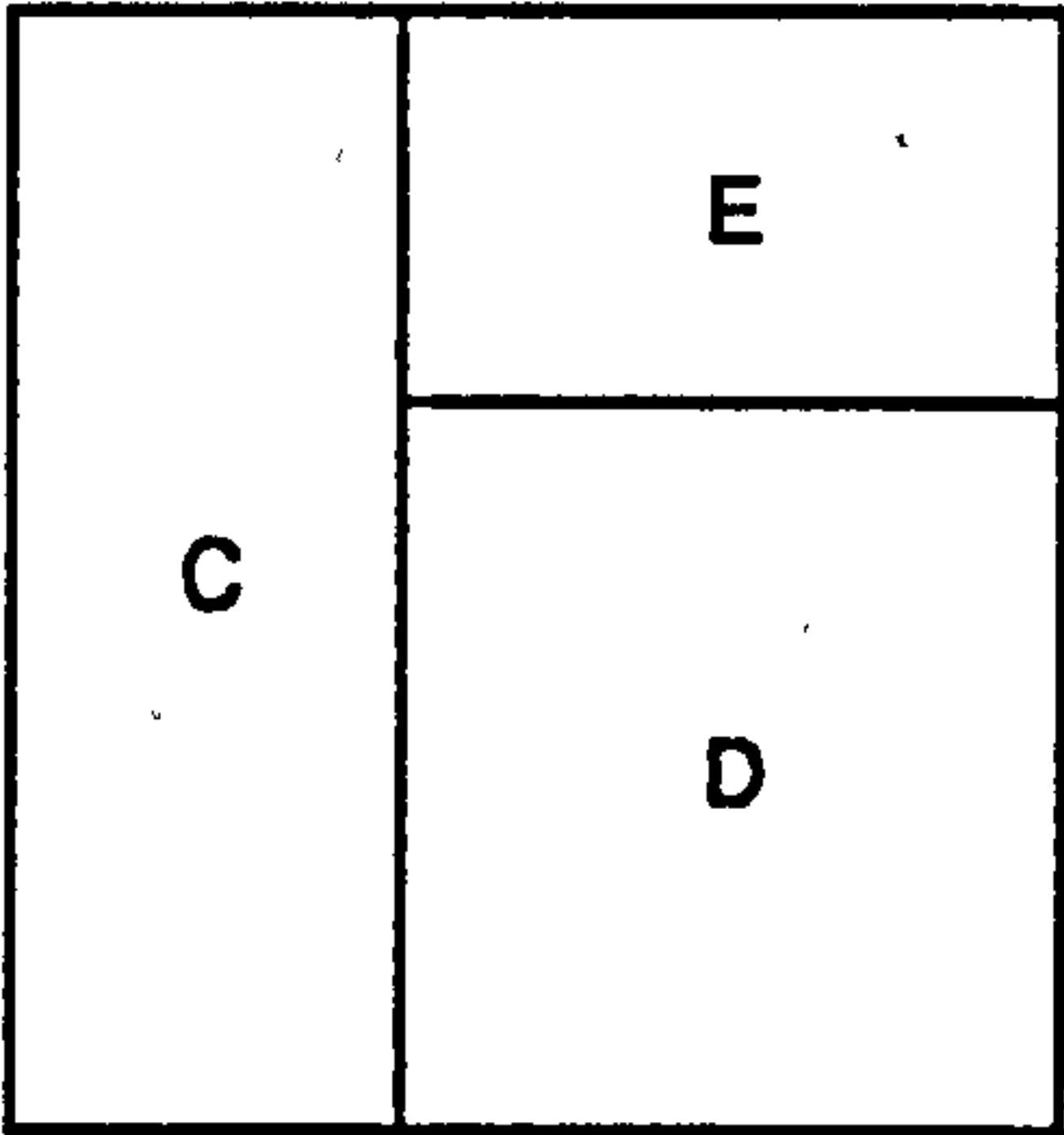
The steps of the process are now described when considering one ceiling face and one floor face:

1. A simple check on the minimum and maximum extents of the faces in the xy plane establishes the possibility of an overlap.
2. A planar graph representing the ceiling face is created by copying its edges. A flag is assigned to each face:

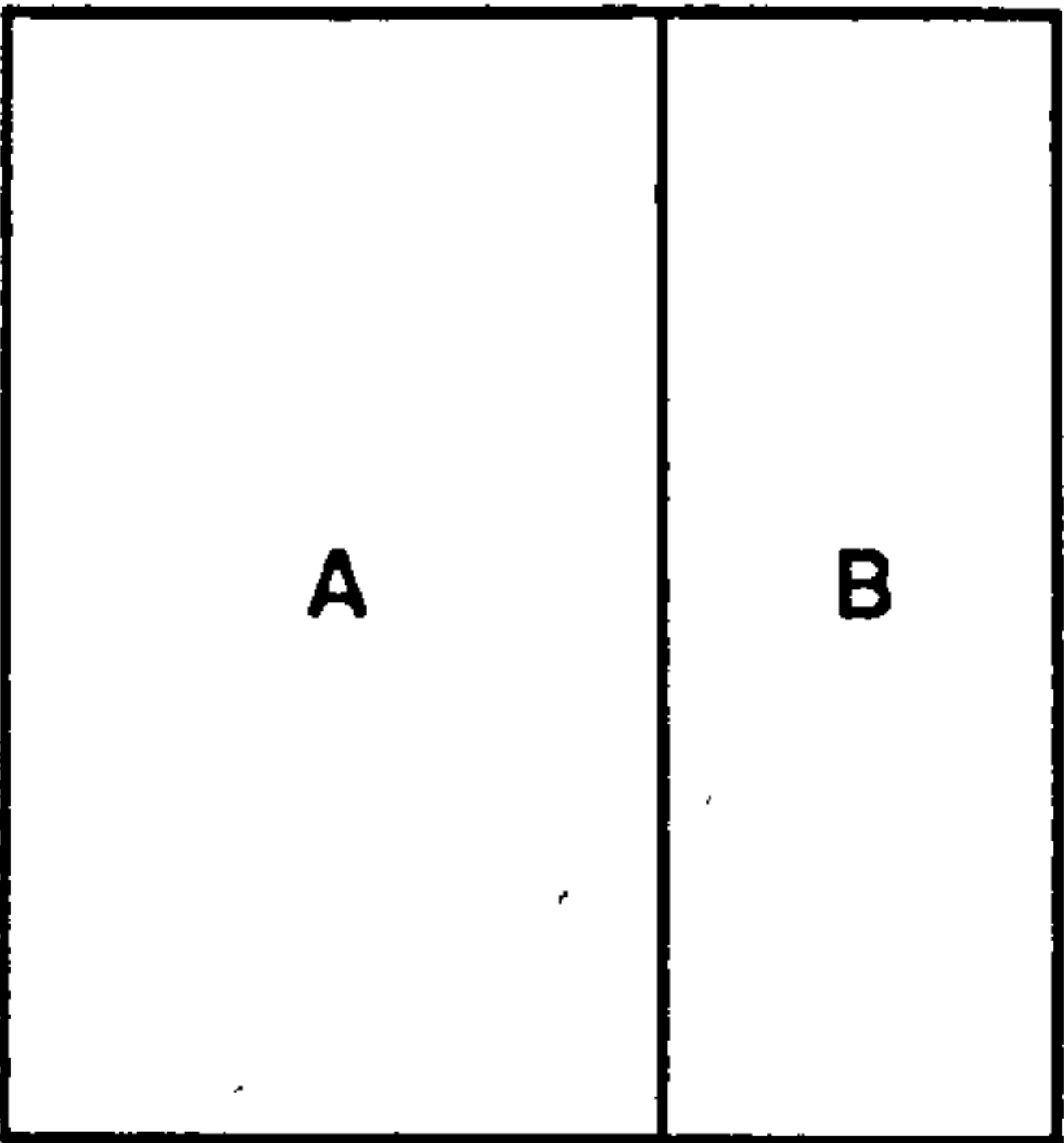
- 1 = original ceiling face.
- 0 = external face, inner faces within hole loops of ceiling face.



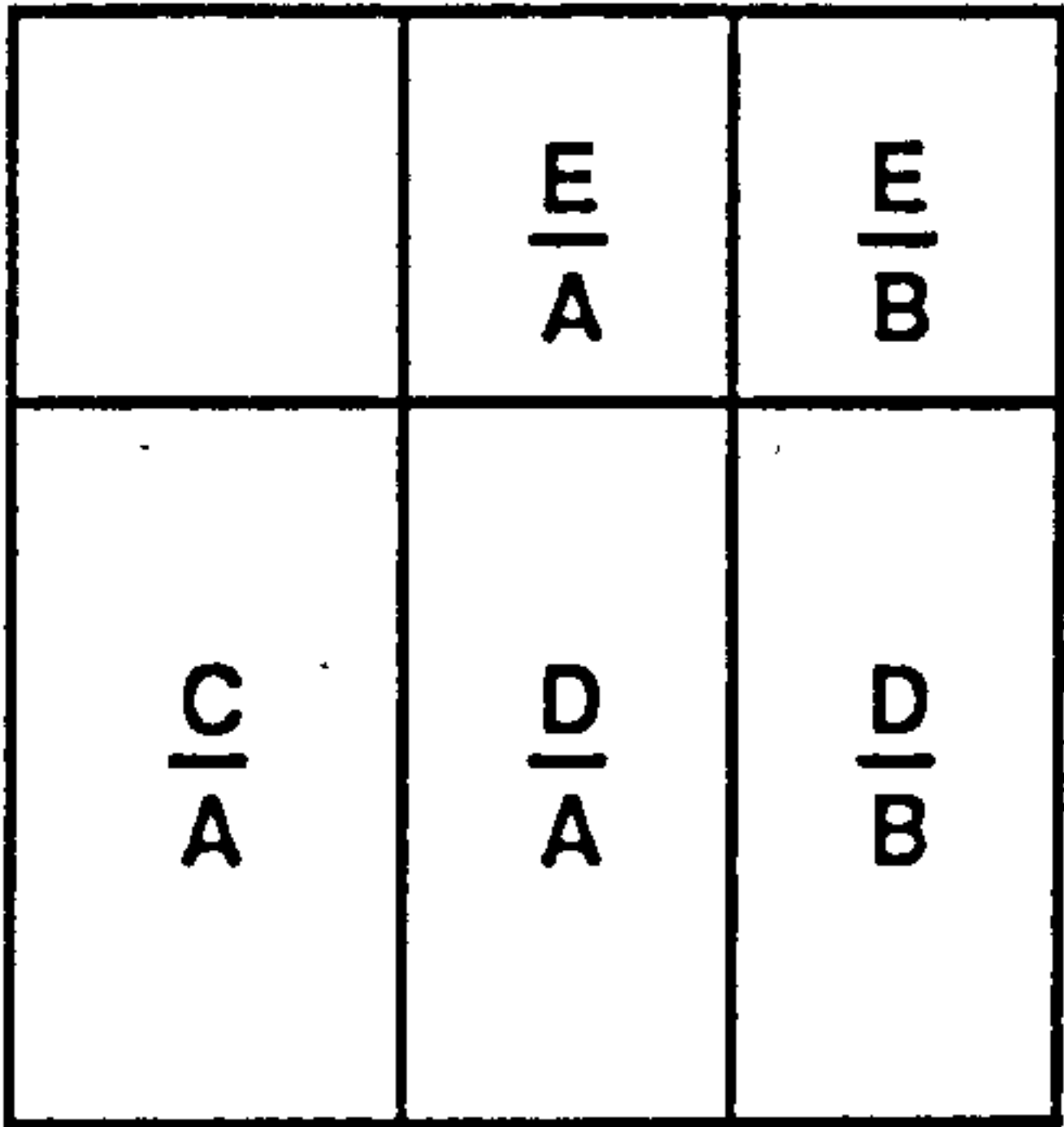




Plan of spaces  
of floor 2



Plan of spaces  
of floor 1



Floor face of space  
Ceiling face of space

**Figure 10-2**   **Constructions created from floor faces of upper floor  
and ceiling faces of lower floor**

3. The edges of the floor face are 'copied' onto the 2D graph, using the same process described in Chapter 8. As the edges are copied the face flags are adjusted so that the resulting planar graph has faces flagged as follows:

- 1 = original ceiling face not overlapped by floor face.
- 0 = external face, inner faces within hole loops of ceiling face or floor face not overlapping ceiling face.
- 1 = floor face overlapping ceiling face.

The flags are adjusted in the following cases:

a) When an edge is added that creates a new face also. See figure 10-3

b) When, as the result of creating a new face, an inner loop is moved between faces, to allow for:

- i) a ceiling face enclosed by a floor face
- ii) a ceiling face enclosed by an inner loop of a floor face. See figure 10-4.

c) When an edge of the floor face is found to be co-incident with an edge of the ceiling face, to allow for

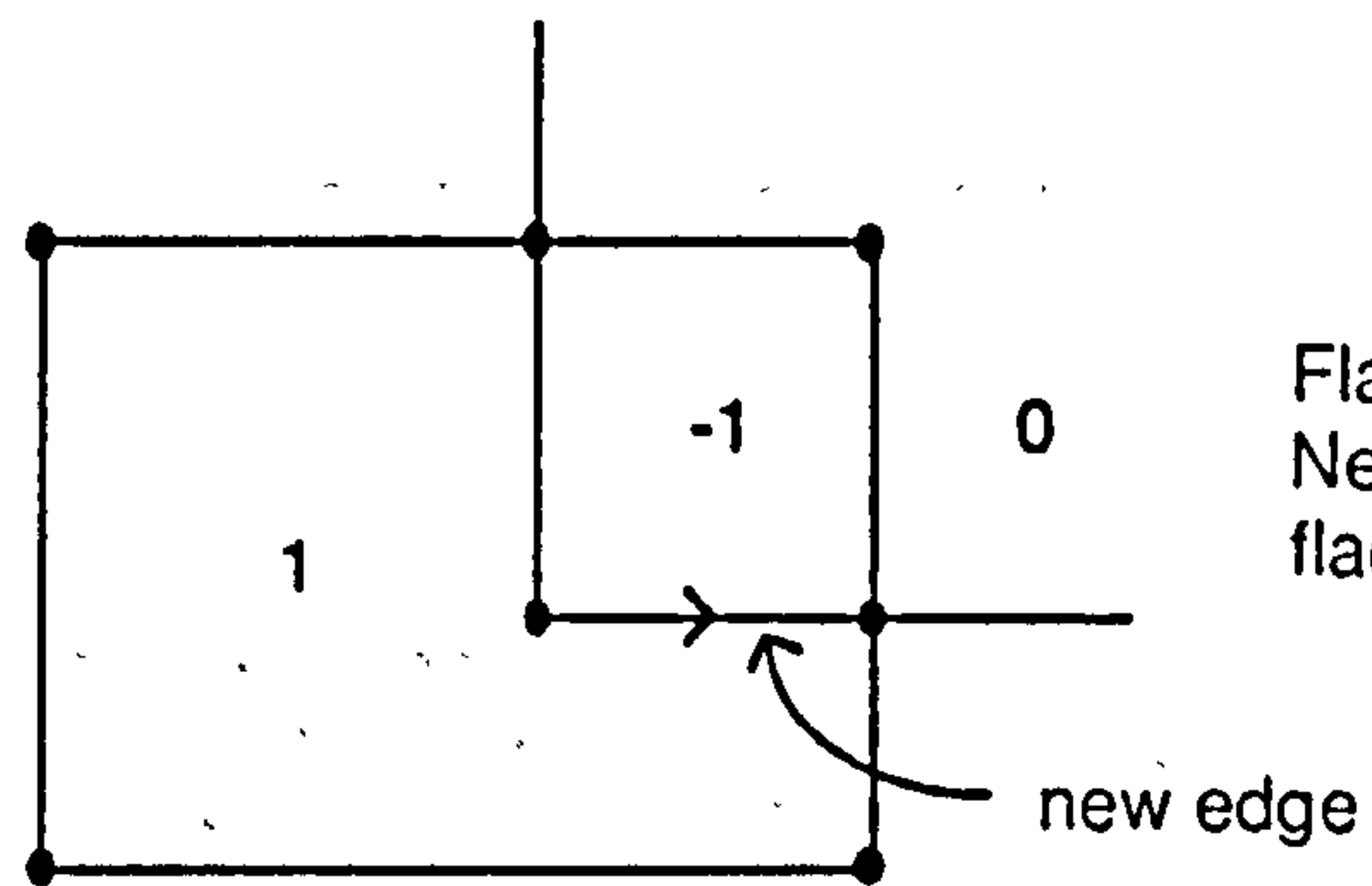
- i) a ceiling face enclosed by a floor face, but with co-incident edges
- ii) a ceiling face enclosed by an inner loop of a floor face, but with co-incident edges. See figure 10-5.

4. The existence of a floor/ceiling construction is therefore indicated by each overlapping face. There may be more than one such face, if either of the overlapping faces are concave in shape. The remainder of the ceiling face, indicated by a flag of 1, is used as the starting face(s) for comparison with the next floor face.

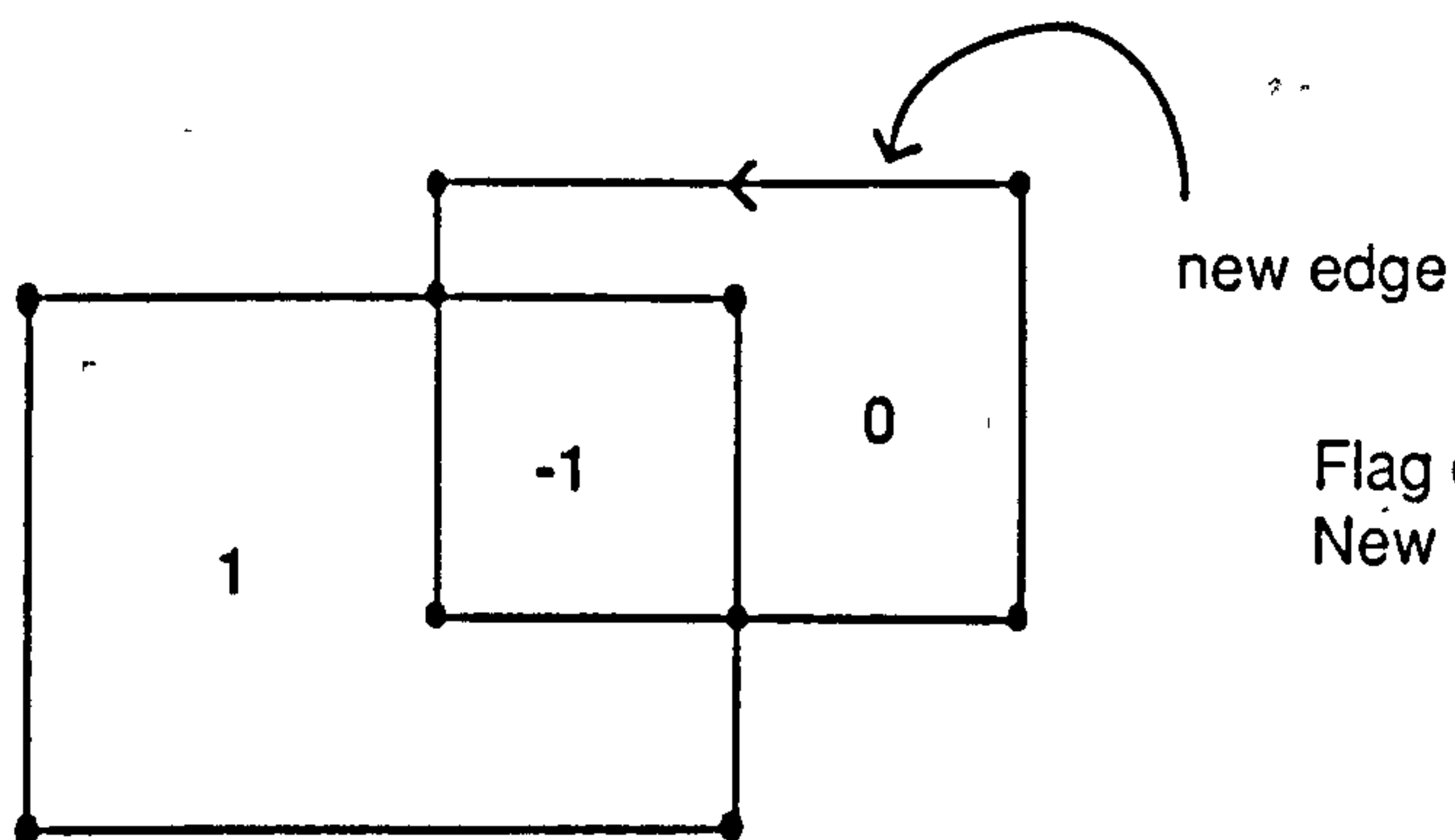
Once the boundaries of the floor/ceiling constructions are known, the second part of the process effects the 'join' between the two existing floor models to create one building model. This requires that the space floor and ceiling faces are divided into more faces according to the defined boundaries, by the addition of edges, and then a construction is created from each matching 'floor' and 'ceiling' face.

The process for every overlapping face as defined above is as follows:

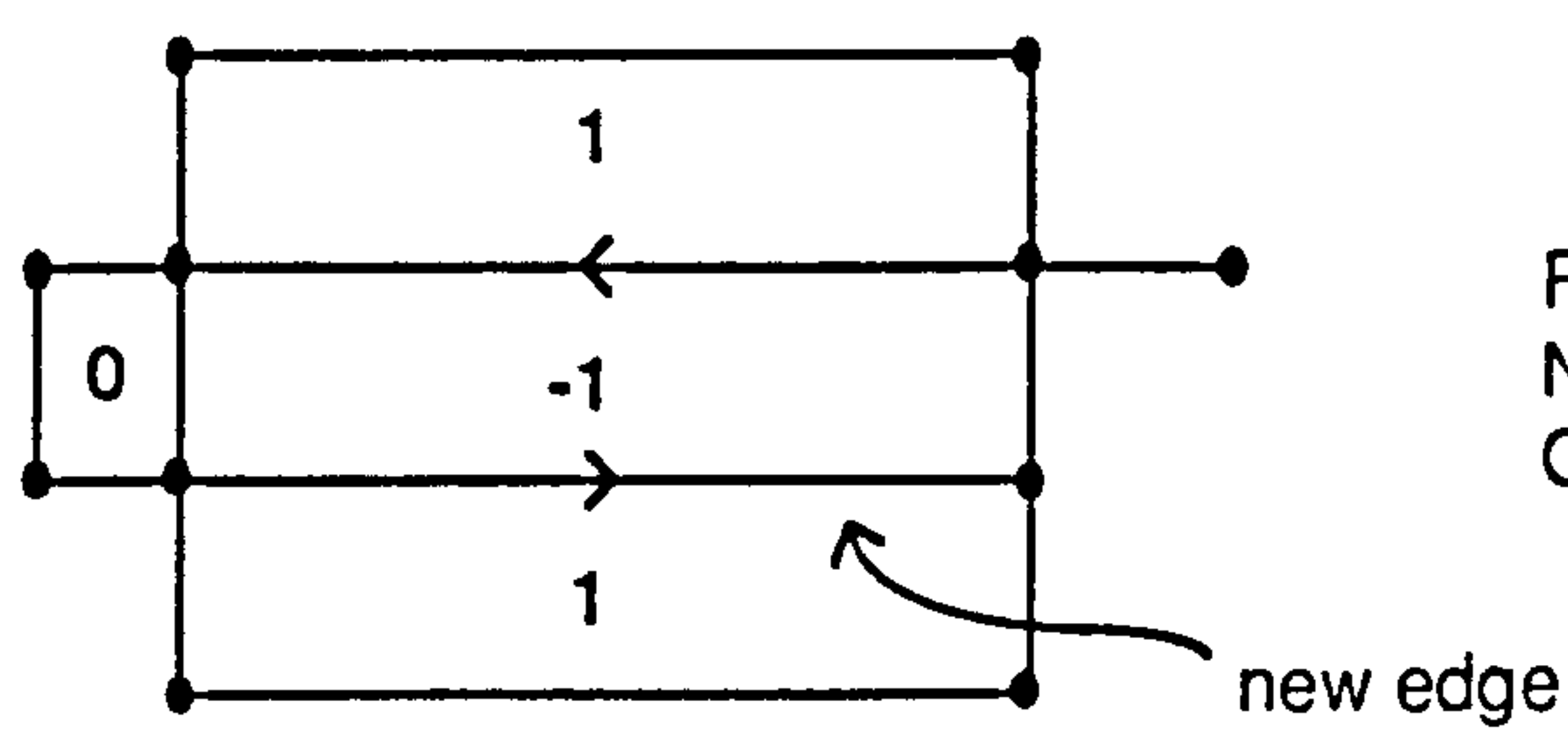
1. For each edge of the boundary of the overlapping face, check the existence of a coincident edge of the space ceiling face. If no edge exists, add an edge with a 'make edge, vertex' operation, or 'make edge, face' operation. The face created with this latter operation becomes the 'ceiling' face of the resulting construction. Existing edges may need to be split to create end vertices of new edges.



Flag of divided face originally 1.  
New face (to left of edge):  
flag set to -1.

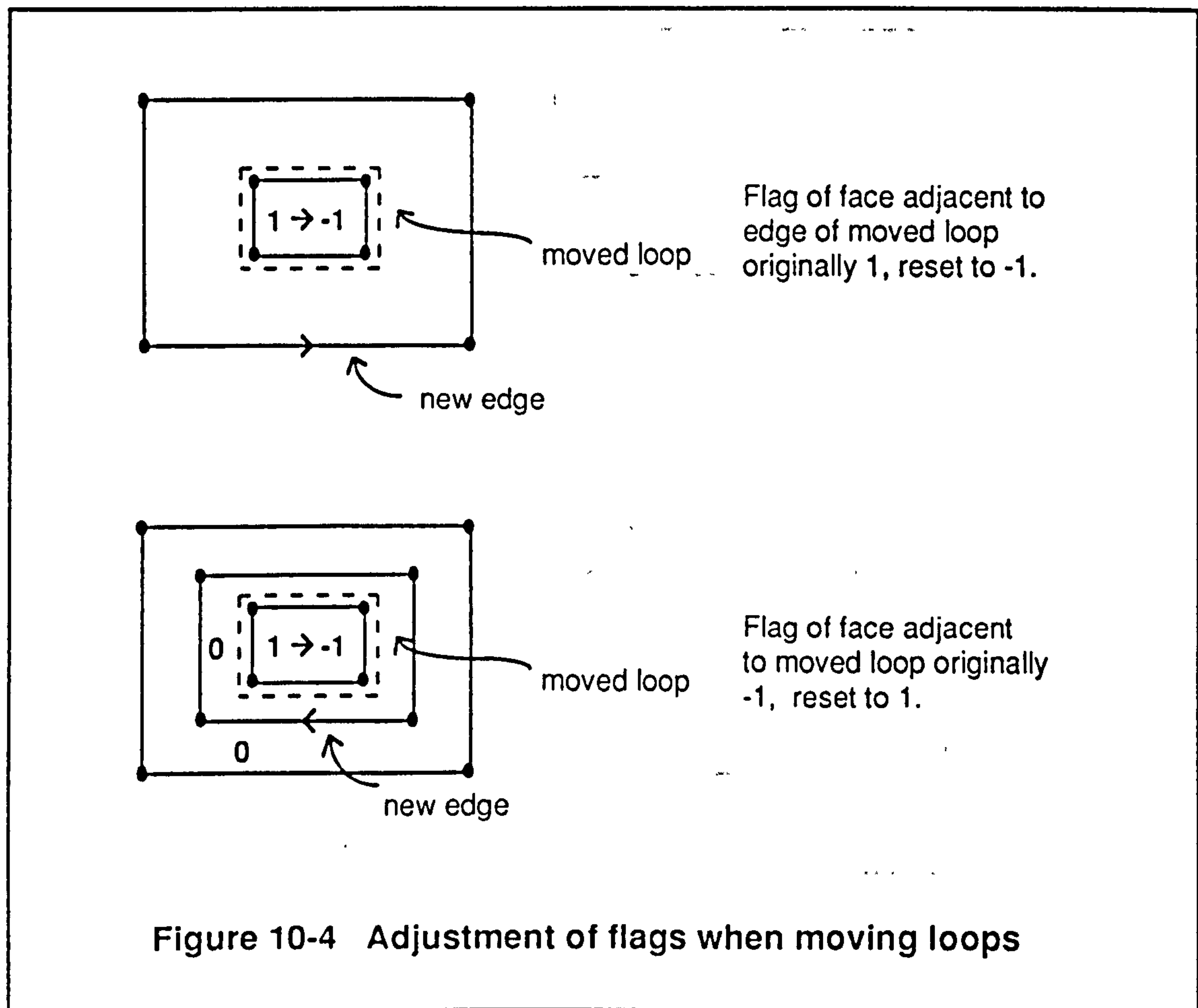


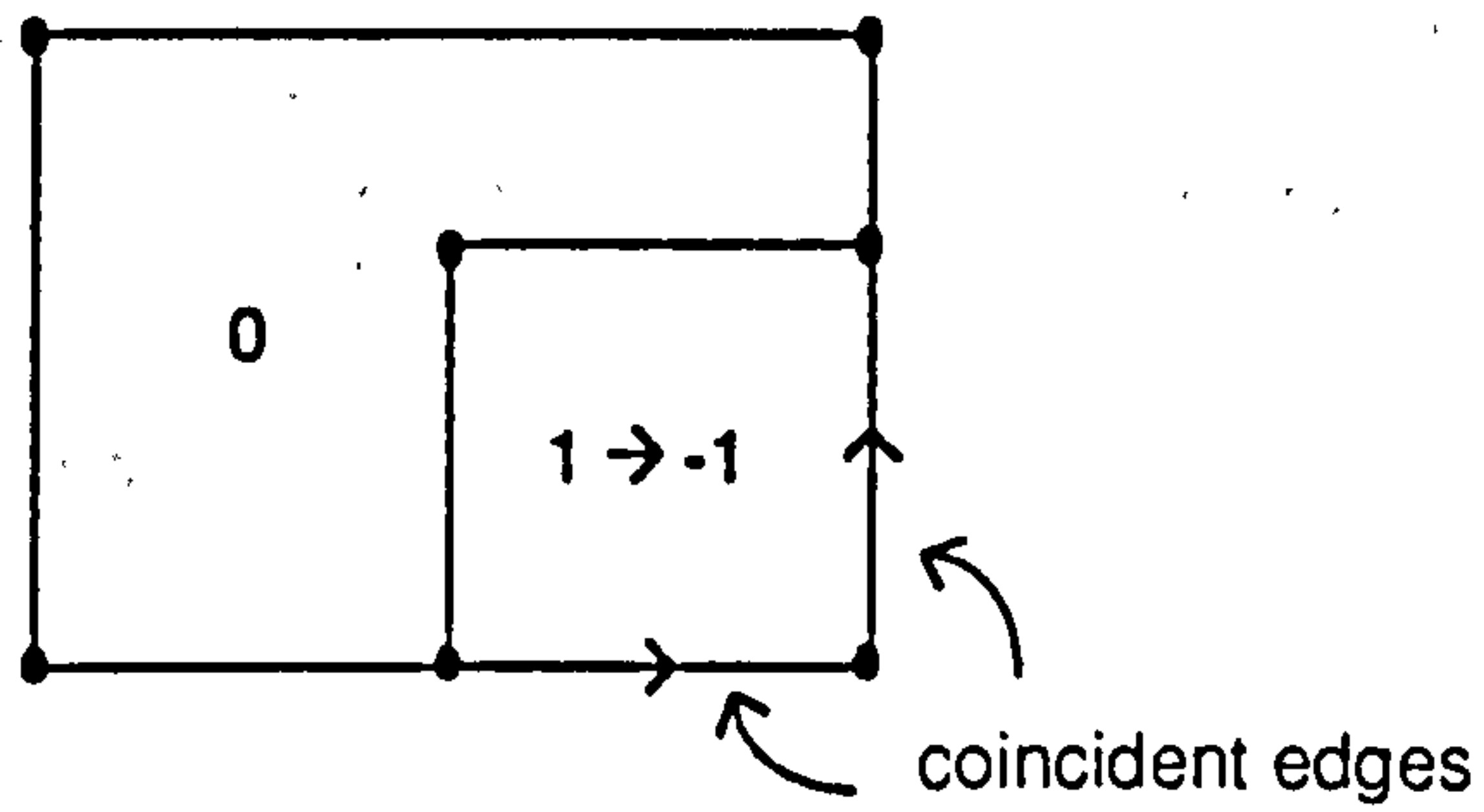
Flag of divided face originally 0.  
New face: flag set to 0.



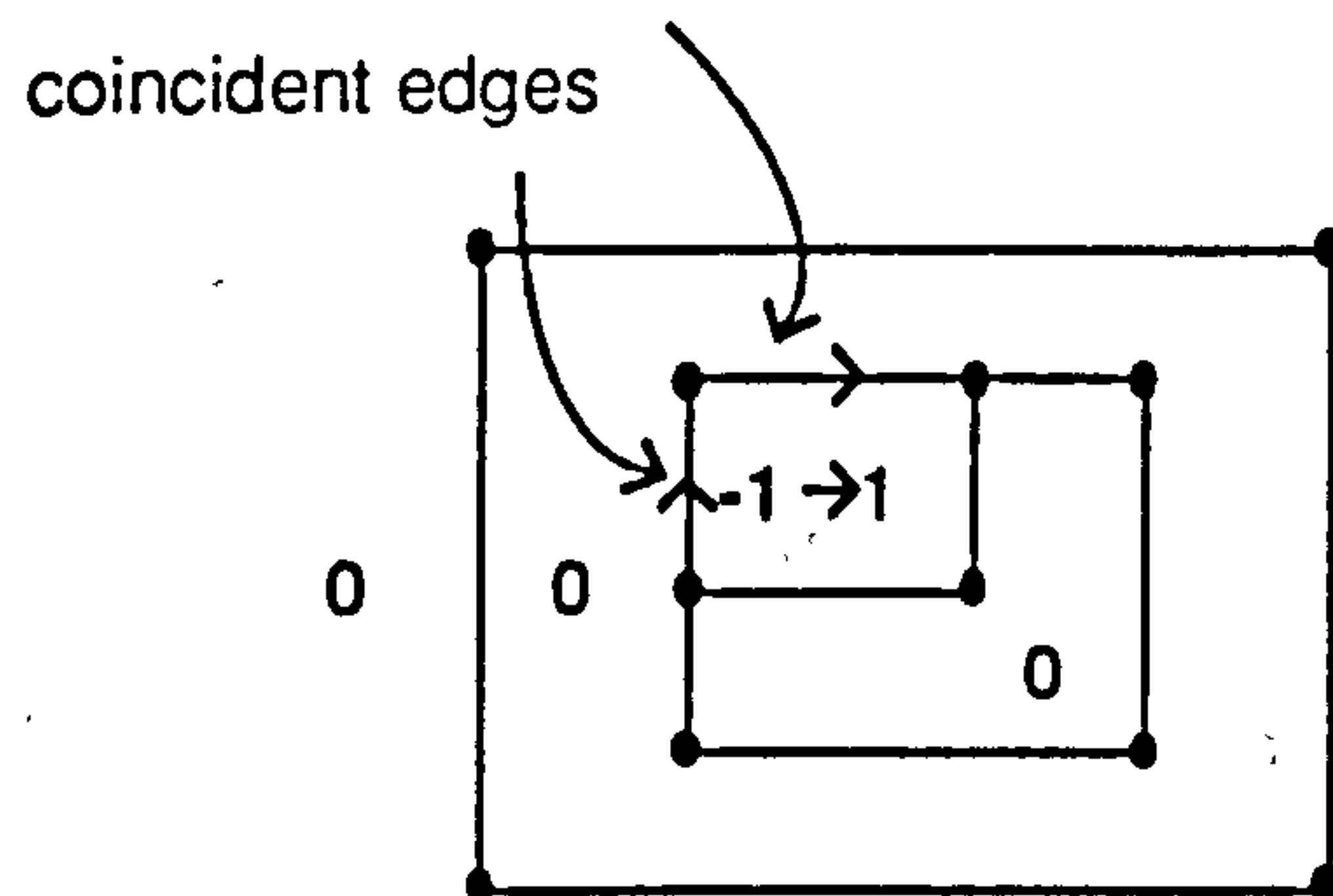
Flag of divided face originally -1.  
New face : flag set to -1.  
Original face: flag set to 1.

Figure 10-3 Adjustment of flags on creation of new face





Flag of face to left of coincident edge is 1, reset to -1.



Flag of face to right of coincident edge is -1, reset to 1.

Figure 10-5 Adjustment of flags when edges are coincident



2. If a new vertex is created from a 'split-edge' or a 'make edge, vertex' operation, a node instance record is also created i.e. the start of a new node. References to the node instance of each vertex of the new face, both existing and new are temporarily saved.
3. Steps 1 and 2 are repeated for the floor face, when different edges and vertices may be created. The face created is the 'floor' face of the resulting construction.
4. For each vertex of the new floor face, a single node is created by merging its node indicated by the node instance, whether new or old, and the node of its corresponding vertex of the new ceiling face.
5. Finally a construction is created referencing the two new faces above, i.e. one floor and one ceiling face.

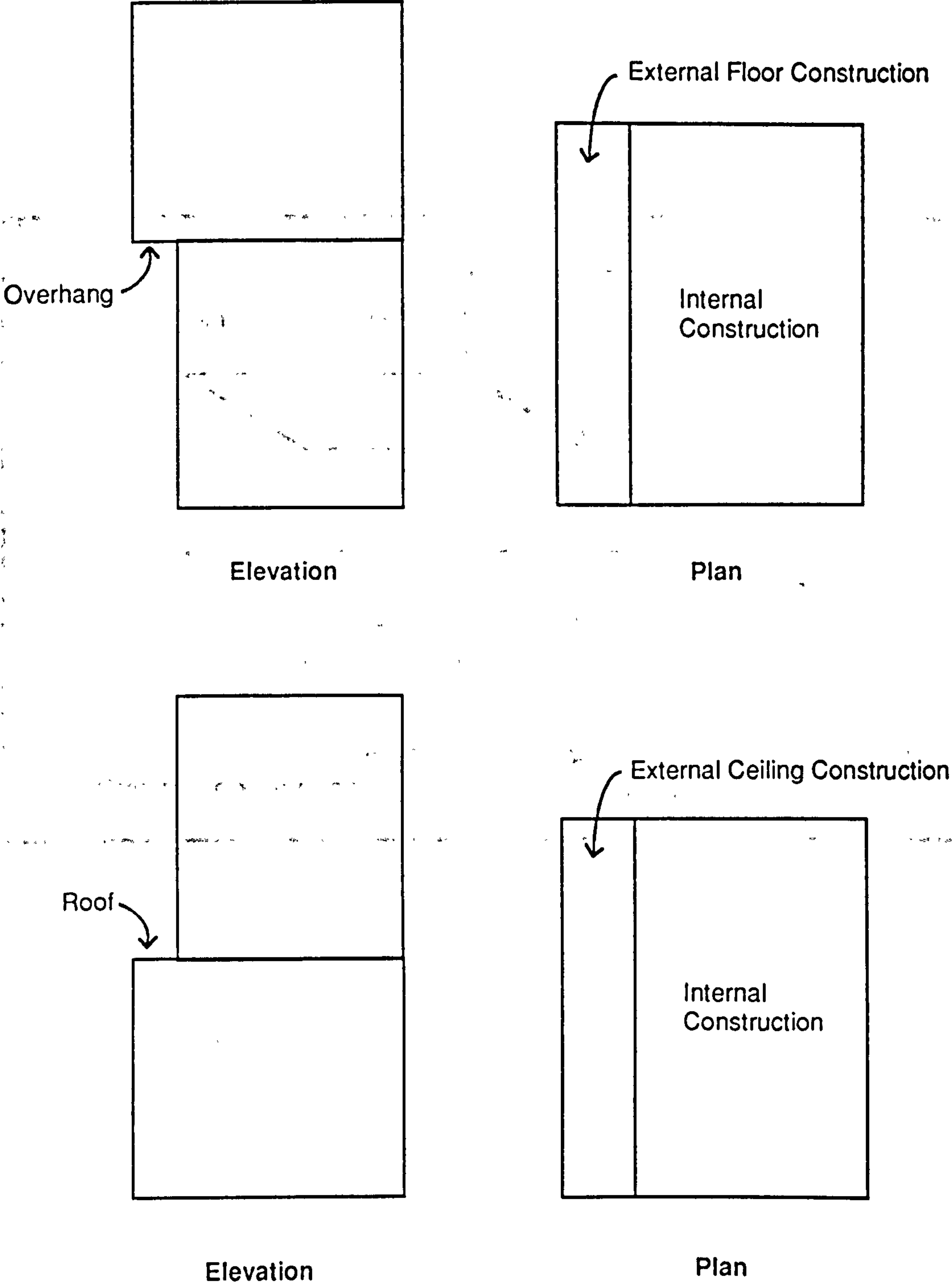
The final part of the process searches for all 'unused' floor and ceiling faces, i.e. those not referenced by a construction. These are the 'external' faces, and for each one a construction is created referencing just the one face. The remaining floor faces represent an 'overhang', i.e. the upper floor is greater in dimension than the lower. The remaining ceiling faces represent roof areas, where the upper floor is less in dimension than the lower floor. See figure 10-6.

## 10.2 Spaces with complex 3D geometry

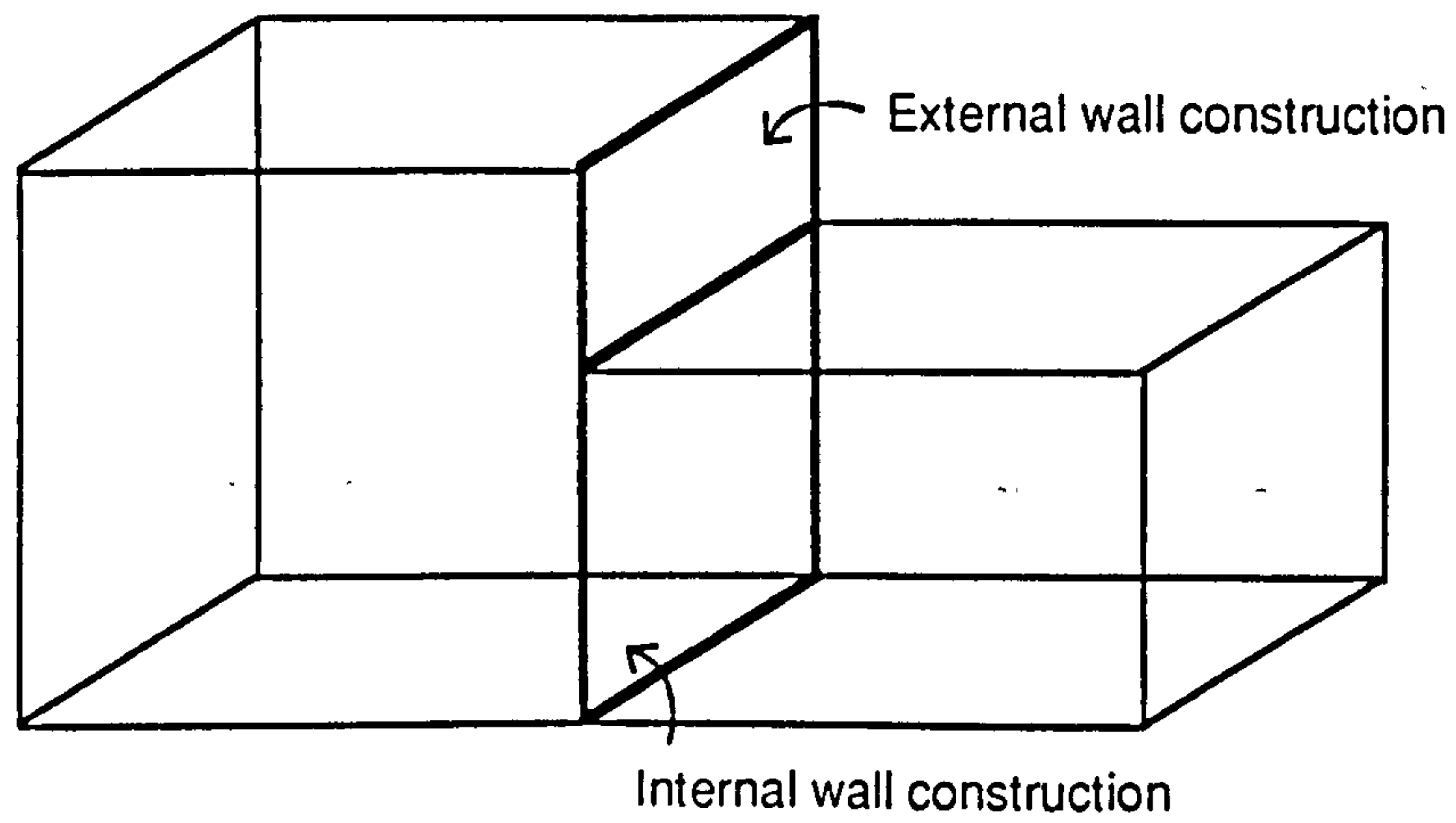
The modelling so far described has been based on the assumption that buildings are essentially  $2\frac{1}{2}$  dimensional. However it is necessary to be able to model buildings with more complex 3 dimensional geometric forms. The requirements are discussed below in order of increasing complexity, followed by a description of the processes involved in creating this geometry.

### 10.2.1 Spaces with varying heights

In order to model spaces of the same floor that have differing floor to ceiling heights it is necessary to assign a different sweep height to each graph face. The sweep process to create the winged-edge structure of one space is therefore unchanged. However, two wall constructions will need to be created from a graph edge that has adjacent faces with different sweep heights: one internal construction, and one external construction, the latter referring to a face of the higher space. Therefore, the face of the higher space resulting from the swept edge must be divided into two faces by a horizontal edge to form the boundary between the internal and external construction. See figure 10-7.



**Figure 10-6** Creation of constructions from remaining external floor and ceiling faces



**Figure 10-7 Division of face of higher space  
to create an internal and external wall construction**

### 10.2.2 Courtyards

A courtyard corresponds to a graph face which is not swept. No space data needs to be created. Therefore the wall constructions created from the edges adjacent to such a graph face will only refer to one space face, as created from the opposite graph face. These wall constructions are then categorised automatically as external, as are those on the perimeter of the building.

The only requirement is, therefore, that such graph faces are identified during the sweep process. These graph faces can be identified as 'external' graph faces in addition to the true external face of the planar graph. note that a graph edge that is adjacent to two 'external' faces does not represent a valid wall construction, either where the faces have been identified as 'external', or where one is the true external face and the other has been identified.

### 10.2.3 Sloping Roof Face

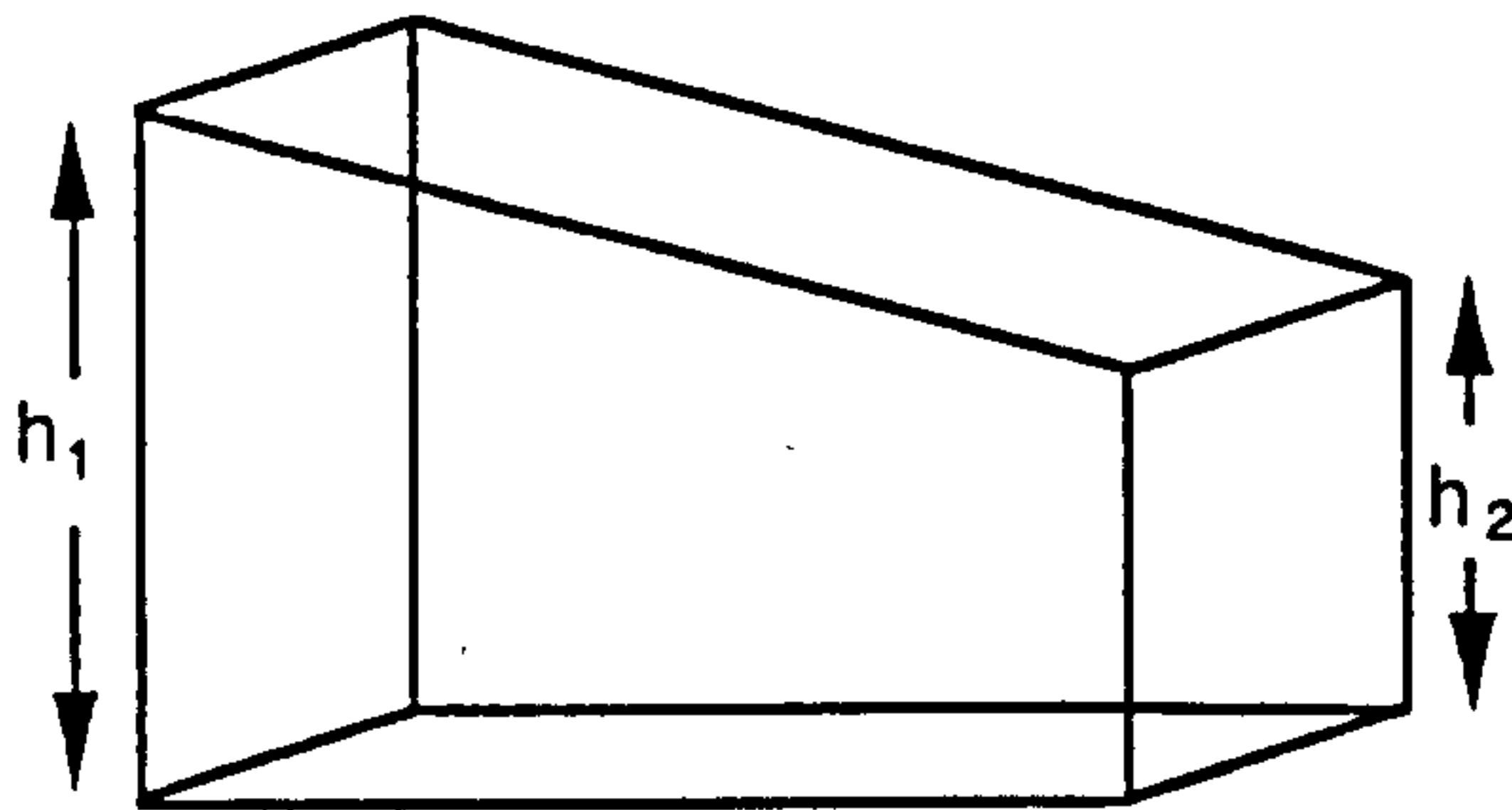
Spaces with sloping roof faces can still be modelled by using the basic method of a sweep operation to create one space. By assigning a different sweep height to each graph vertex with respect to each graph face, an inclined ceiling face can be created. See figure 10-8. The resulting topology of the sweep operation of one space is unchanged, but non-rectangular side faces are created.

User interface options are necessary to define these vertex or space heights. Therefore checks are required to ensure that the resulting ceiling face is planar.

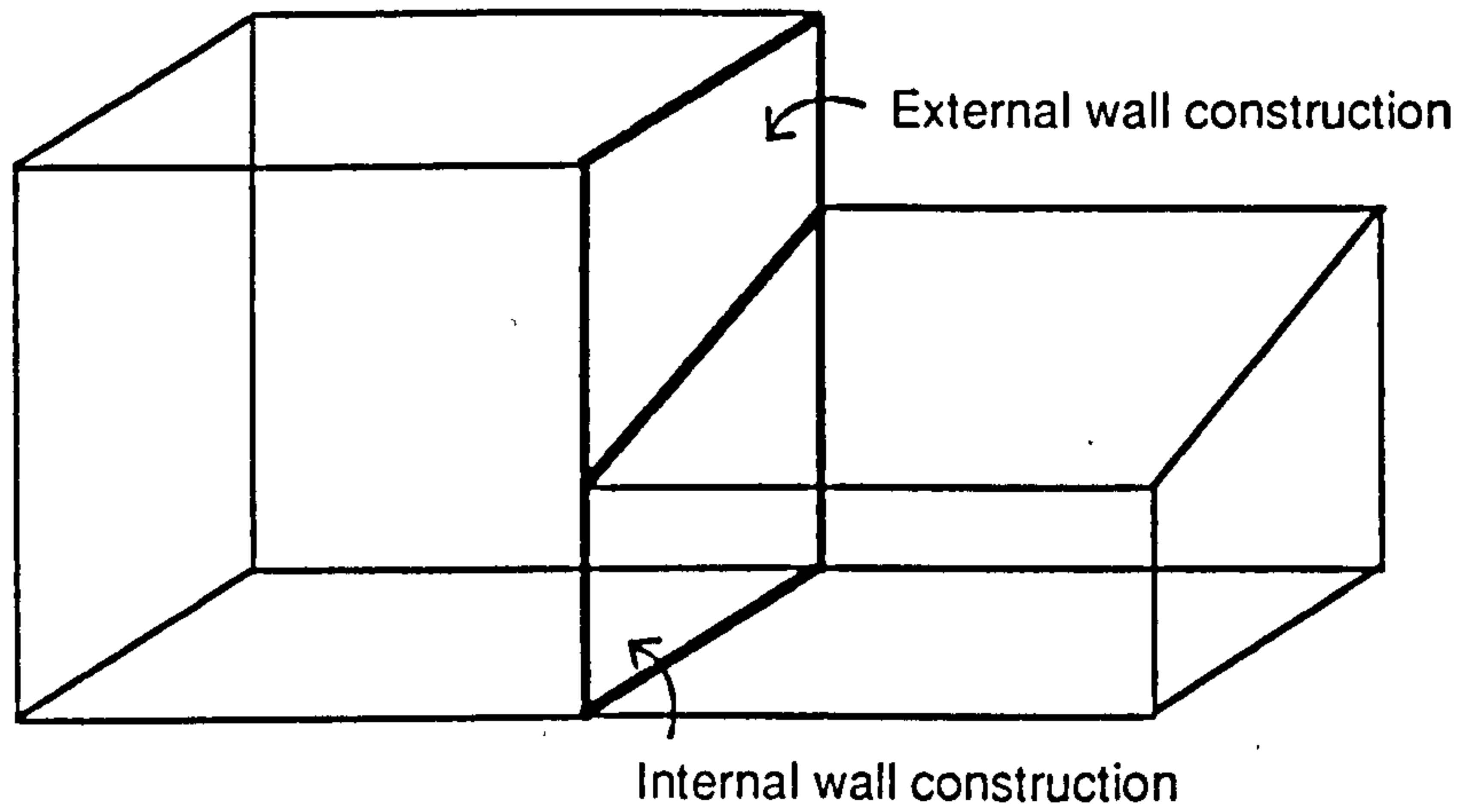
As above an internal and external wall construction may be created from one graph edge. In this case the original swept side face of the 'higher' space must be divided by a non-horizontal edge. See figure 10-9. A case of further complexity is shown in figure 10-10, where an internal and two external constructions are created, each face being divided by a non-horizontal edge.

### 10.2.4 Spaces with more than one sloping face

It is necessary to be able to model spaces such as those shown in figure 10-11. This can be achieved again by modifying the basic sweep operation, with the assignment of a different sweep direction to each vertex. This can be assigned by the user in the form of an angle of inclination of the wall face. It can be ensured that the resulting topology of one space is still unchanged from the basic shape by checking that the side faces do not intersect other than at the 'up' edges, resulting from sweeping a graph vertex.

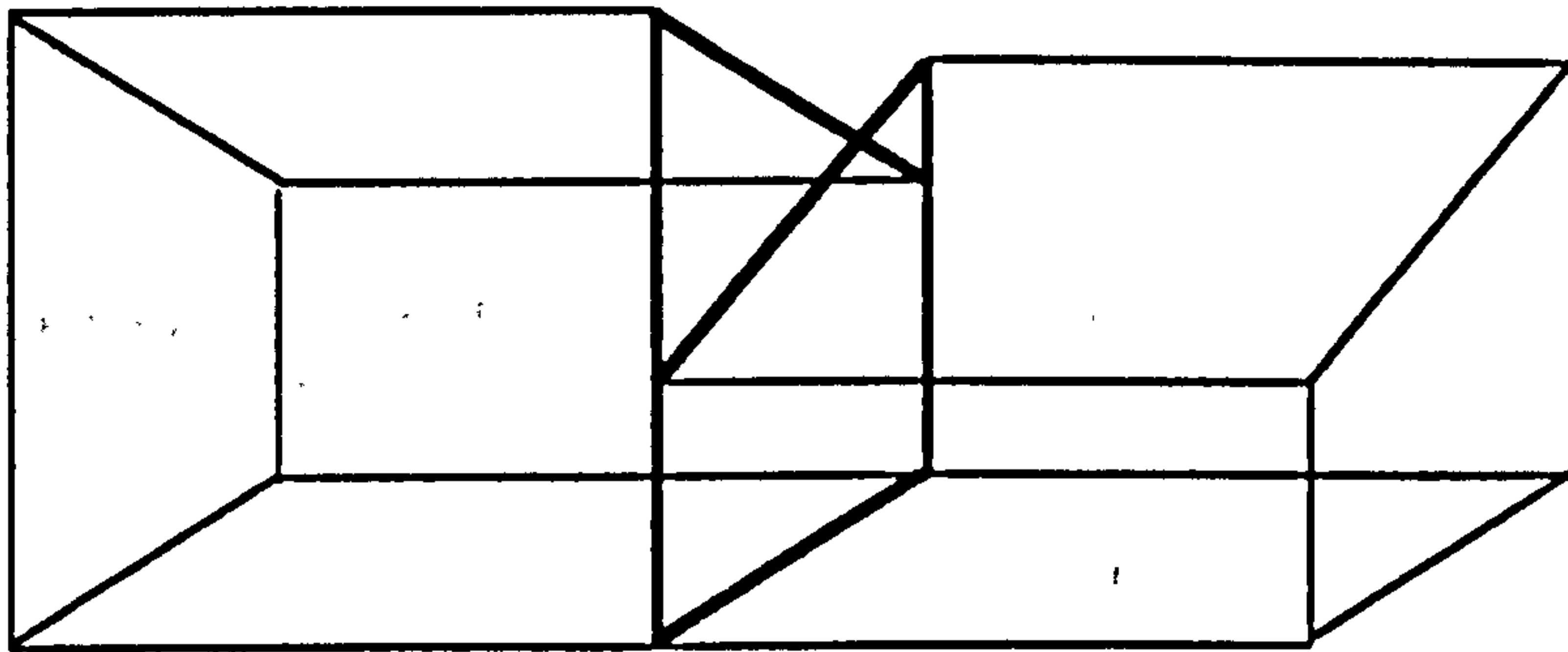


**Figure 10-8** Creation of inclined ceiling face, by using different sweep heights for each vertex

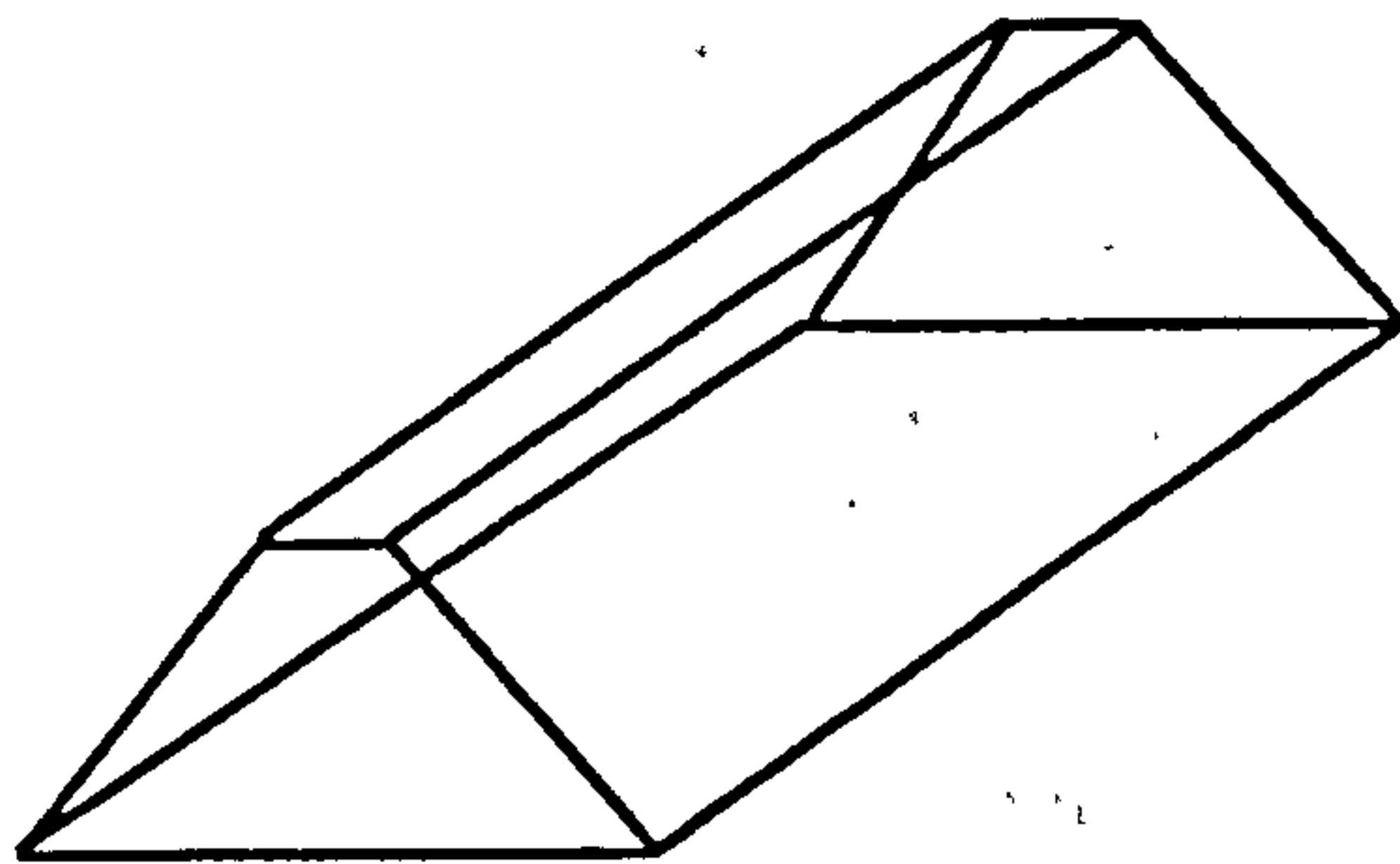


**Figure 10-9** Division of face by non-horizontal edge





**Figure 10-10 Division of both faces by edges**



**Figure 10-11 Space with more than one sloping face**

This complicates further the creation of wall constructions, and the division of side faces into the internal and external wall constructions. To maintain consistency, it is desirable that an internal construction is always created from an internal graph edge, therefore the assignment of an inclination should only be allowed for external walls, i.e. those resulting from sweeping an external graph edge.

#### 10.2.5 'Merged' Spaces

More complex 3 dimensional forms can be created without deviating from the basic sweep operation that creates a similar topology for each space, by 'merging' spaces together. A roof void shaped as shown in figure 10-12 can be modelled by merging two spaces as created by the standard process i.e. by removing the internal wall construction. It can be seen that the 2 dimensional plan of the shape can be drawn exactly as if the wall was present. A 'delete wall construction and space' operation is then required to merge the spaces.

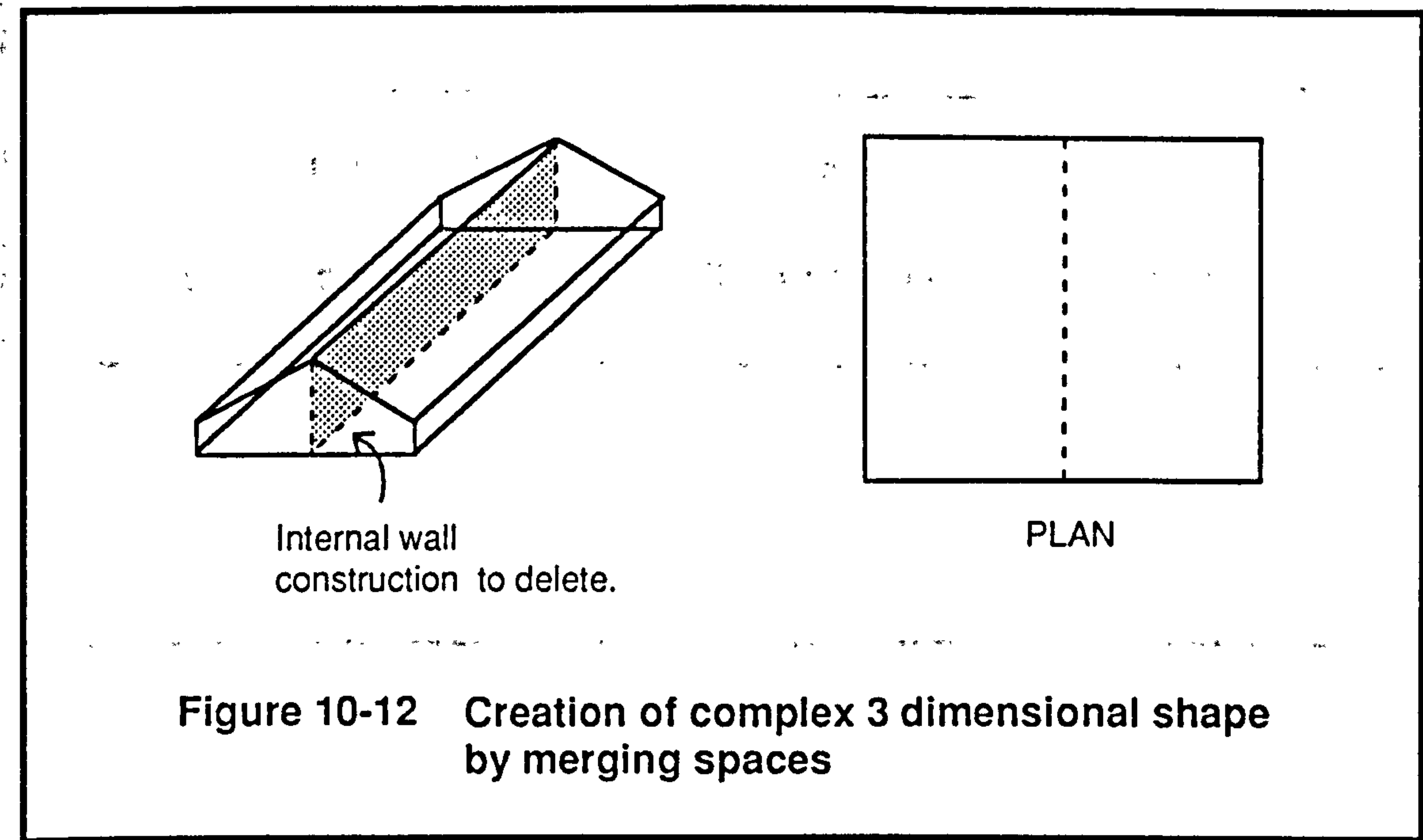
#### 10.2.6 Spaces extending over more than one floor

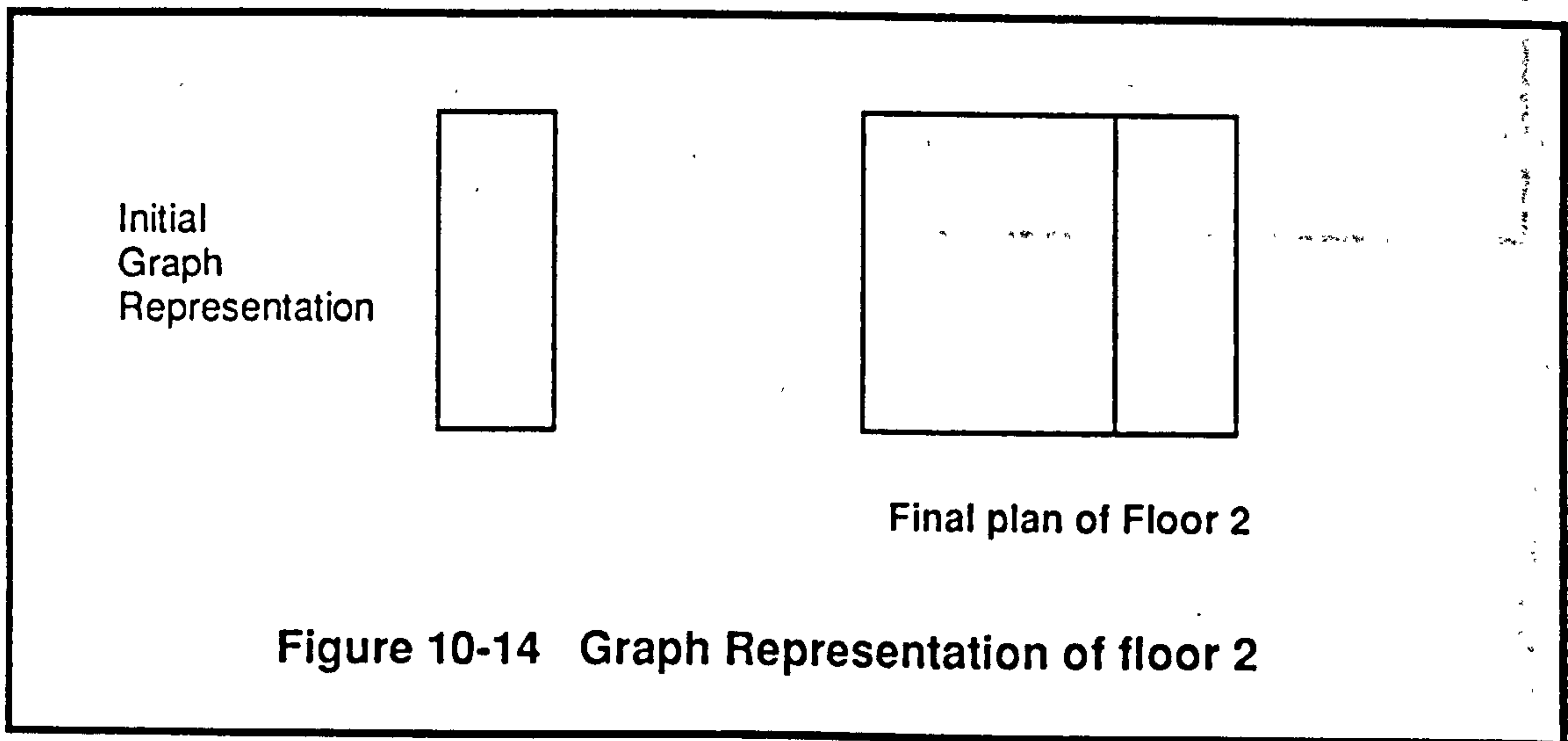
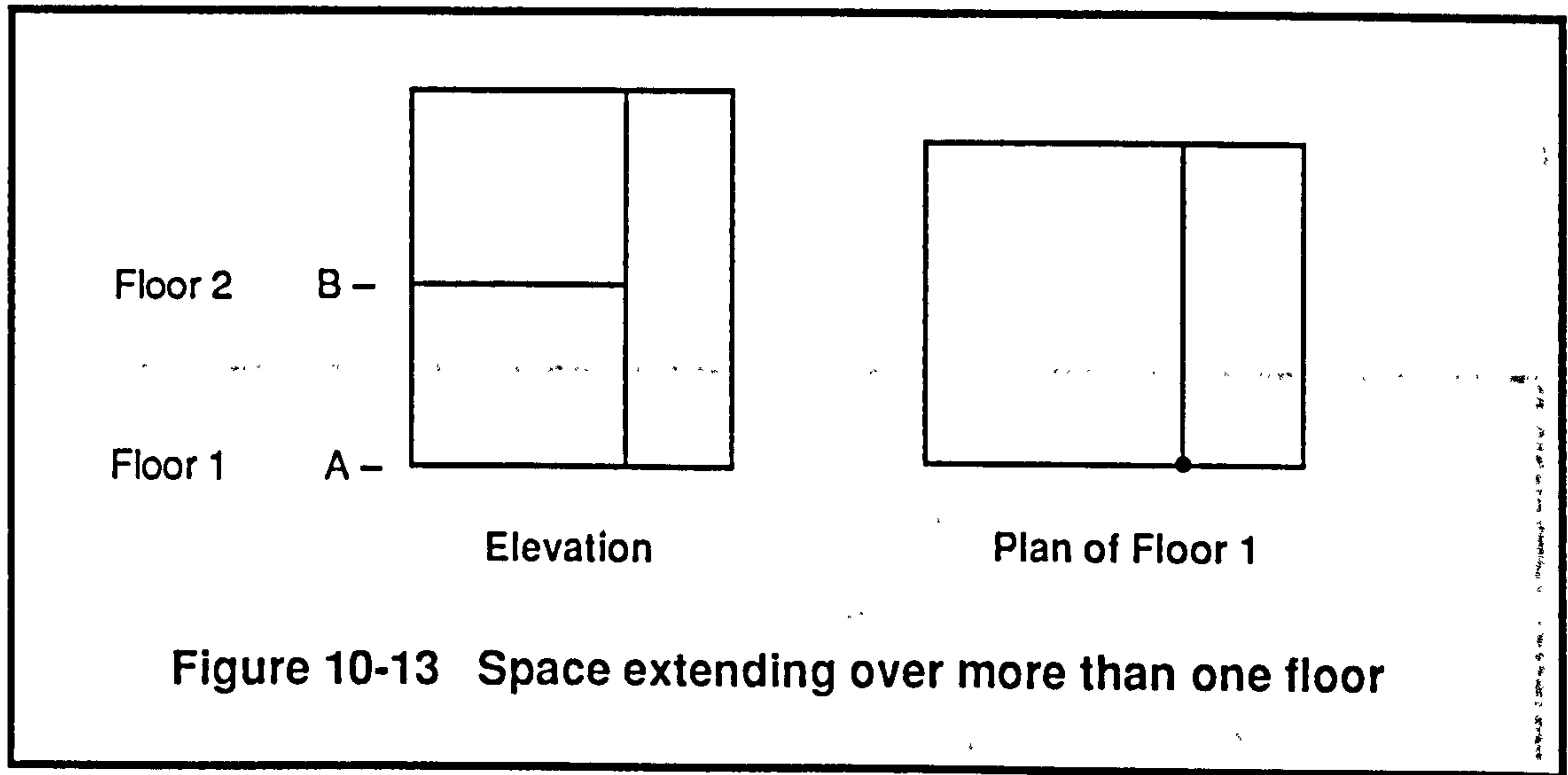
The above modelling techniques also need to be considered in terms of multi-floor buildings. So far, this complex 3 dimensional geometry has only been considered on a floor by floor basis. As discussed in 10.1, it is necessary to consider the process of merging floors of a building, and a process was described which assumed that all of the spaces of one floor had the same floor to floor height.

It can be assumed that all sloping ceiling faces are external, i.e. spaces of upper floors are only positioned on top of spaces with horizontal ceiling faces. However varying space heights allow spaces to be defined which extend over more than one floor. See figure 10-13.

Floor 1 is defined at height A, and floor 2 is to be defined at height B. The height of a space of floor 1 is greater than B, e.g. an entrance hall. After the spaces of floor 1 have been created, the higher space will have side faces referenced by internal constructions at the lower level, and external constructions at the upper floor level. The addition of adjacent spaces at the upper floor level to this space must result in these external constructions becoming internal constructions.

Therefore, in order to be able to define these adjacent spaces, instead of starting the input of floor 2 with no graph edges or faces, an initial graph representation must exist of the extended space. New edges can then be attached to the existing edges to create the faces representing the adjacent spaces. See figure 10-14. Note that it is an invalid operation to add edges that intersect these initial graph faces, in effect adding walls into empty space, and a check is therefore required to prevent this.





It is therefore necessary to be able to access the building model to check for existing spaces on a floor, to create an initial graph representation of these spaces.

A further complication arises if a sloping ceiling face intersects a floor level. See figure 10-15. In this case, the ceiling face must be divided at its intersection with the floor level i.e. two external ceiling constructions are created, and a graph edge must represent this.

### 10.3 Creation of Complex 3D Geometry

The requirements specified in 10.2 have implied various modifications to the basic sweep process. These are now discussed, together with the necessary extensions to the data representation of the building model.

#### 10.3.1 Sweep process using a vertex sweep height and direction

The basic sweep operation described in Chapter 9 assumes that the co-ordinates of a swept vertex  $(x,y,z)$  are  $(x,y,z+h)$ , where  $h$  is the sweep height, and the sweep direction is vertical. Every space vertex resulting from sweeping the same graph vertex has the same co-ordinates.

To satisfy the requirements specified above, it is necessary:

- a) to calculate different sweep co-ordinates for a graph vertex with respect to each of its adjacent graph faces, using different sweep heights and
- b) to allow for a sweep direction determined by the wall inclinations assigned to the two adjacent edges of the vertex with respect to each space.

The calculation of the swept co-ordinates  $(x', y', z')$  from the co-ordinates  $(x,y,z)$  using a sweep height of  $h$ , and with a wall inclination assigned to the two adjacent edges follows. See figure 10-16.



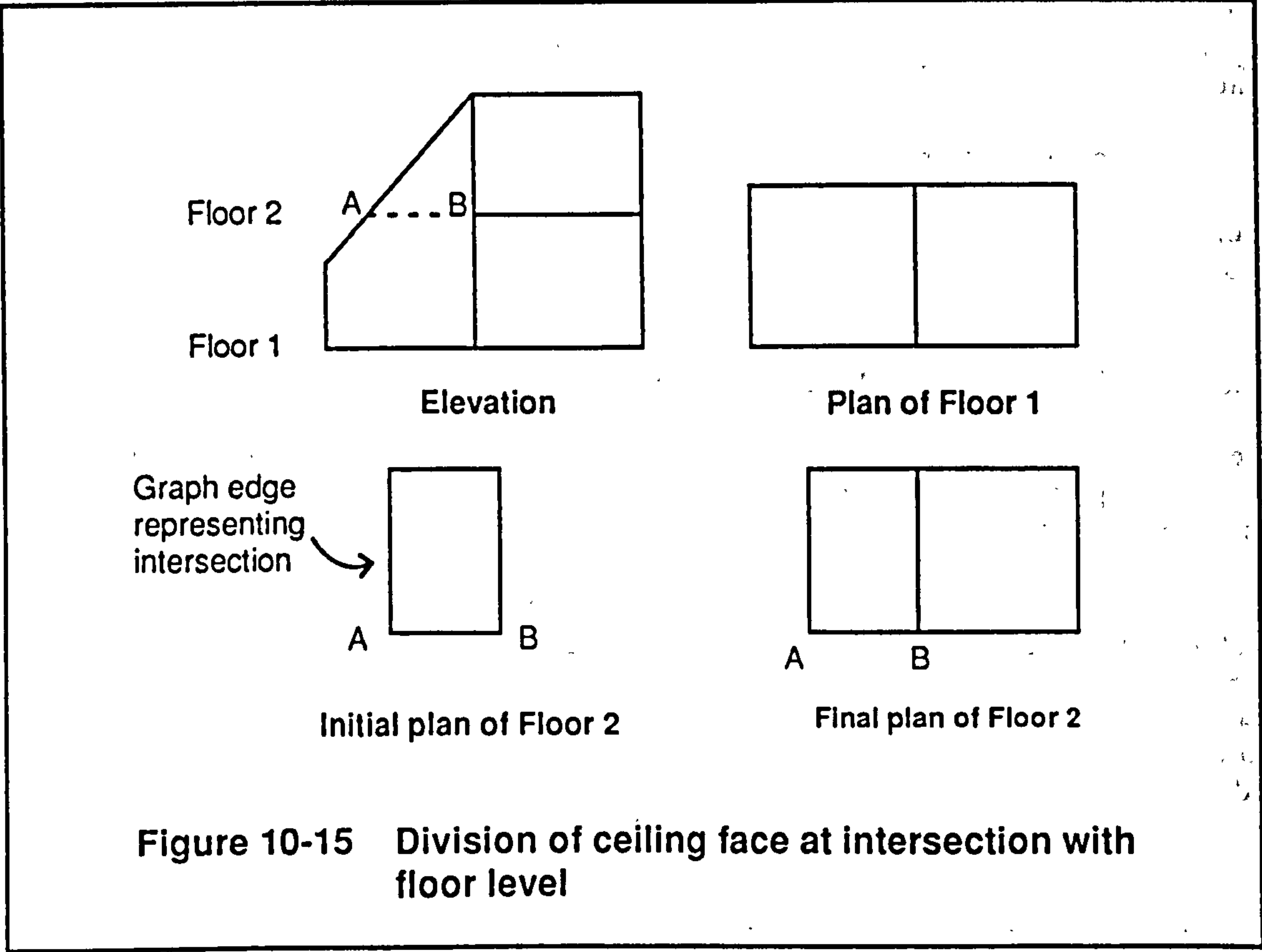


Figure 10-15 Division of ceiling face at intersection with floor level

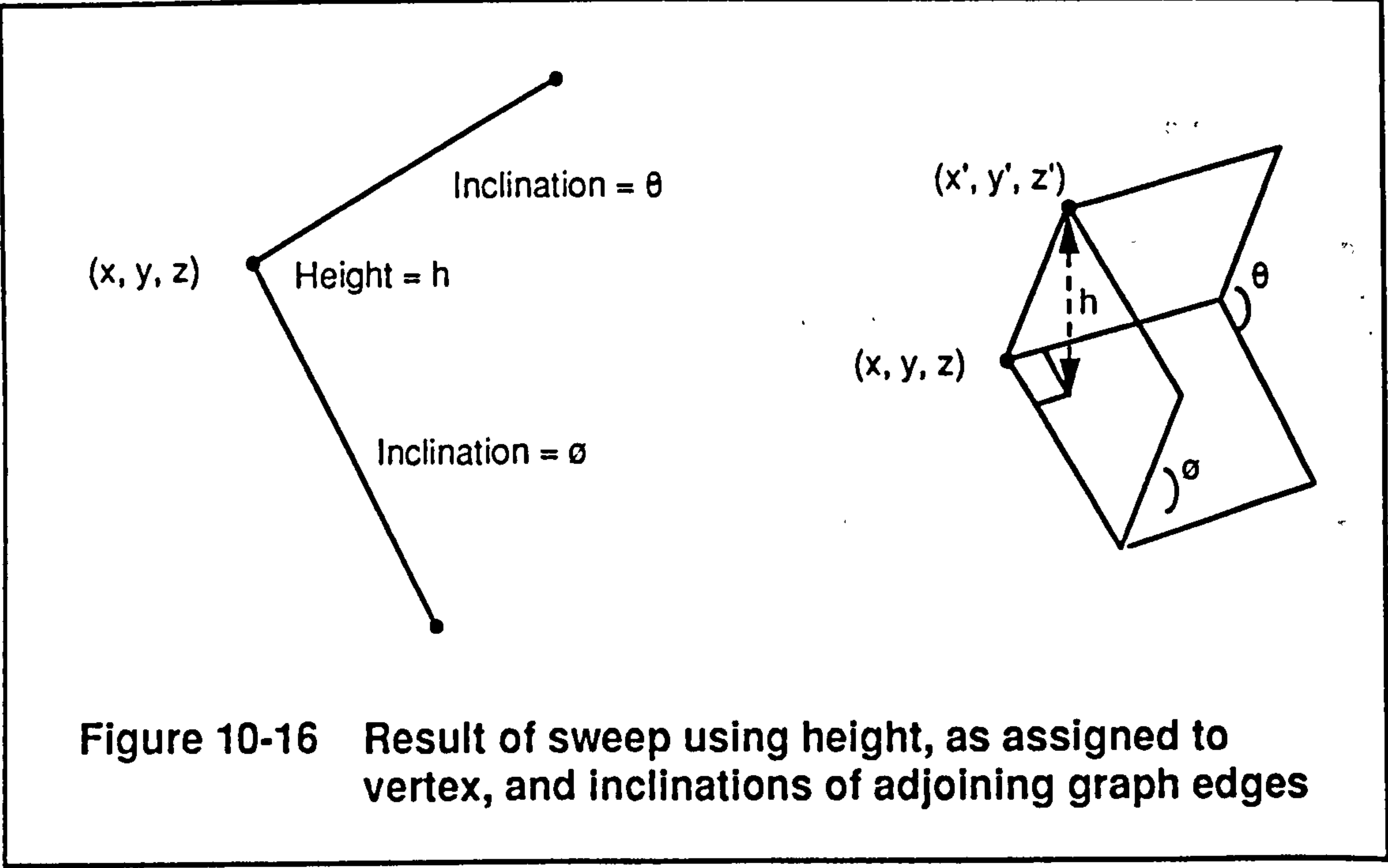


Figure 10-16 Result of sweep using height, as assigned to vertex, and inclinations of adjoining graph edges

1. The normal of the side face resulting from sweeping a graph edge is necessary for the definition of the wall construction. It is now calculated from the direction of an edge  $\underline{u}$  and the assigned inclination angle  $\emptyset$ :

Given  $\underline{u} = a_i + b_j + 0k$

$\underline{v}$  is the perpendicular vector in the xy plane:

$$\underline{v} = b_i - a_j + 0k$$

$\underline{n}$  is the required normal:

$$\underline{n} = \underline{v} \text{ rotated by } (90 - \emptyset) \text{ degrees about } \underline{u}$$

$$\Rightarrow \underline{n} = \cos\emptyset \cdot b_i - \cos\emptyset \cdot a_j - \sin\emptyset \cdot k$$

See figure 10-17.

2. The calculation of the swept point  $\underline{p}'$ , with co-ordinates  $(x', y', z')$  is dependent on the originating point  $\underline{p}$ , the normals calculated from the two adjacent edges  $\underline{n}_1$  and  $\underline{n}_2$  and the sweep height  $h$ :

$$\underline{p}' = \underline{p} + \mu \underline{d}$$

where  $\underline{d}$  is the sweep direction, and  $\mu$  the sweep distance. See figure 10-18.

The sweep direction is the cross product of the normals:

$$\underline{d} = \underline{n}_1 \times \underline{n}_2$$

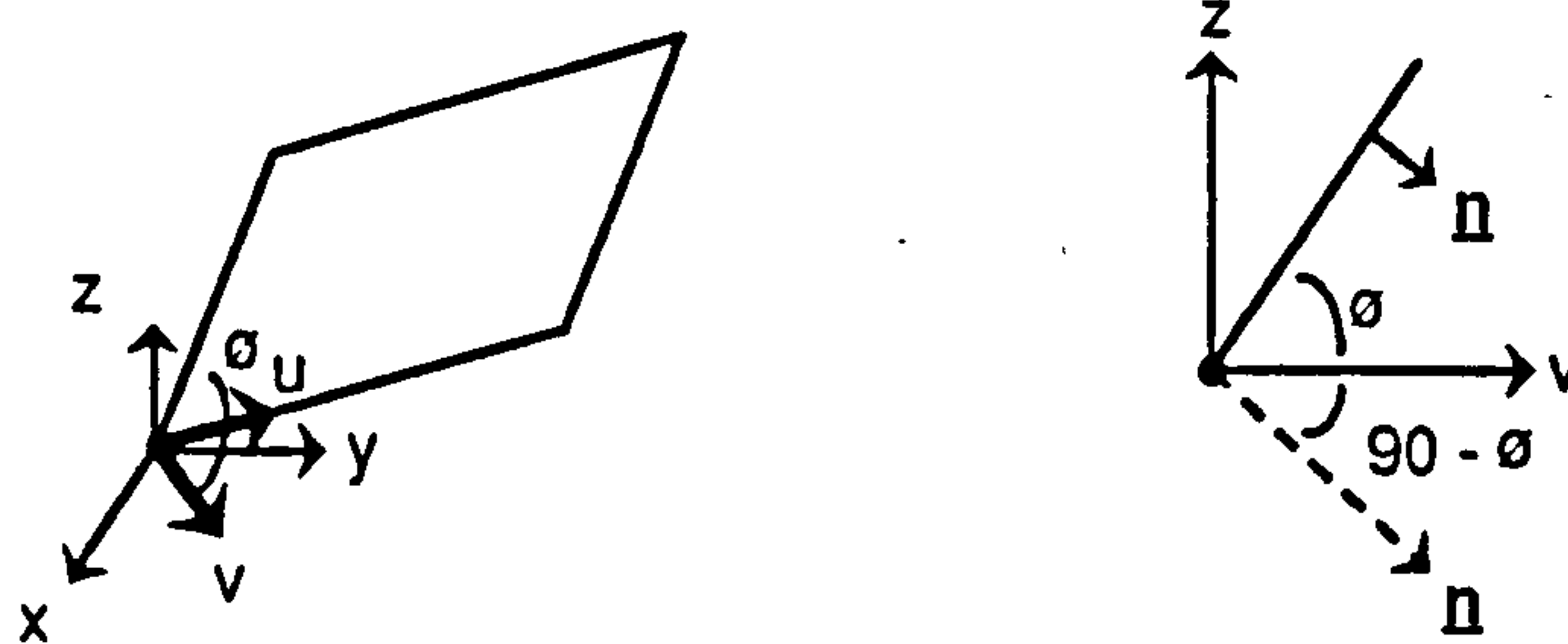
Assuming  $z' = z + h$ , if  $\underline{d} = l_i + m_j + nk$

$$\Rightarrow \mu = h/n$$

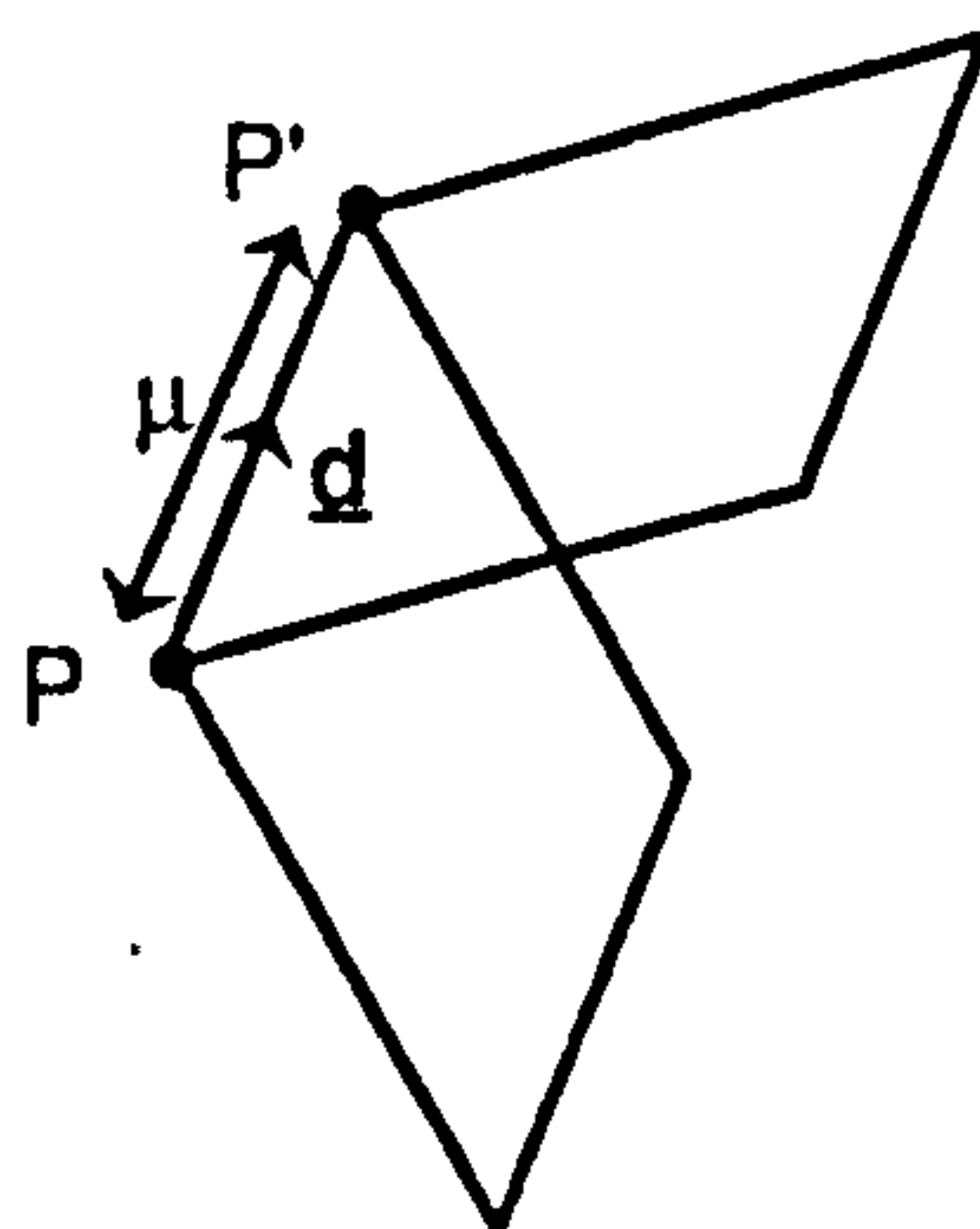
There may be vertices connecting co-linear edges. The calculation for these differs from the above, and their co-ordinates are dependent on the swept co-ordinates of all the 'corner' vertices as above with respect to one graph face. The co-ordinates of the swept corner vertices determine the plane of the ceiling face, and therefore the swept co-ordinates of the 'co-linear' vertices must lie on this plane also, i.e. on the line of intersection of the plane of the side faces created from the co-linear edges and the ceiling plane. See figure 10-19.

Where  $\underline{u}$  is the direction of the co-linear edges, and  $\underline{n}$  is the normal of the side faces, the sweep direction  $\underline{d}$  is the cross product:

$$\underline{d} = \underline{u} \times \underline{n}$$



**Figure 10-17** Calculation of normal of side face from direction of graph edge and assigned inclination



**Figure 10-18** Calculation of swept point

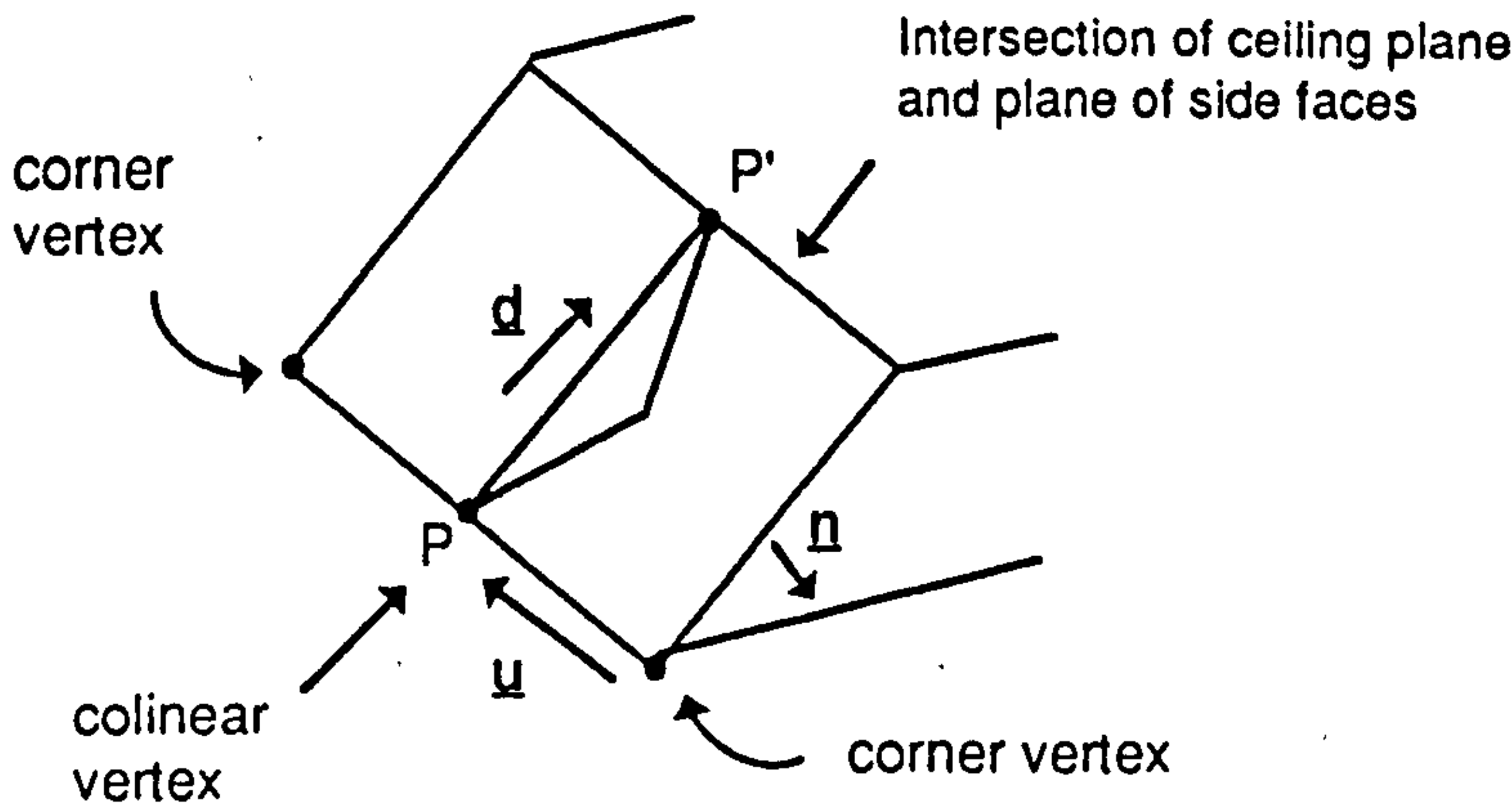


Figure 10-19 Calculation of swept point  $P'$  for a colinear vertex at point  $P$

The swept point  $p'$  is at the intersection of two lines: one through the original point  $p$  with direction  $\underline{d}$ , and the other connects the swept points of the two corner vertices. There is a special case that must be allowed for, where the former line does not intersect the second line segment, but intersects the 'up' edge of the adjacent corner vertex, as shown in figure 10.20. Because of this possibility the sweep operation is modified such that it is completed for one space using all corner vertices, and then the 'up' edges for the colinear vertices are added dividing the 'true' side faces as necessary.

Using the above calculations, there are some checks that must be made, as discussed previously, to ensure that the basic topology resulting from the sweep operation is maintained:

a) Side faces should not intersect other than at the 'up' edges, as shown in figure 10-21. This is ensured if it is checked that each side face is well-formed i.e. that the edges of a side face only intersect at a vertex. A simple check is to calculate the scalar product of the direction vectors of the lower and top edges. If the result is negative, then the vectors are in 'opposite' directions, and the 'up' edges must intersect. A side face that has coincident swept vertices i.e. the top direction vector is the null vector, is allowed. In this case the topology is not modified, an edge exists with zero length.

b) Further checks are required to ensure that the side faces resulting from sweeping edges of inner loops do not intersect with the side faces of outer loops.

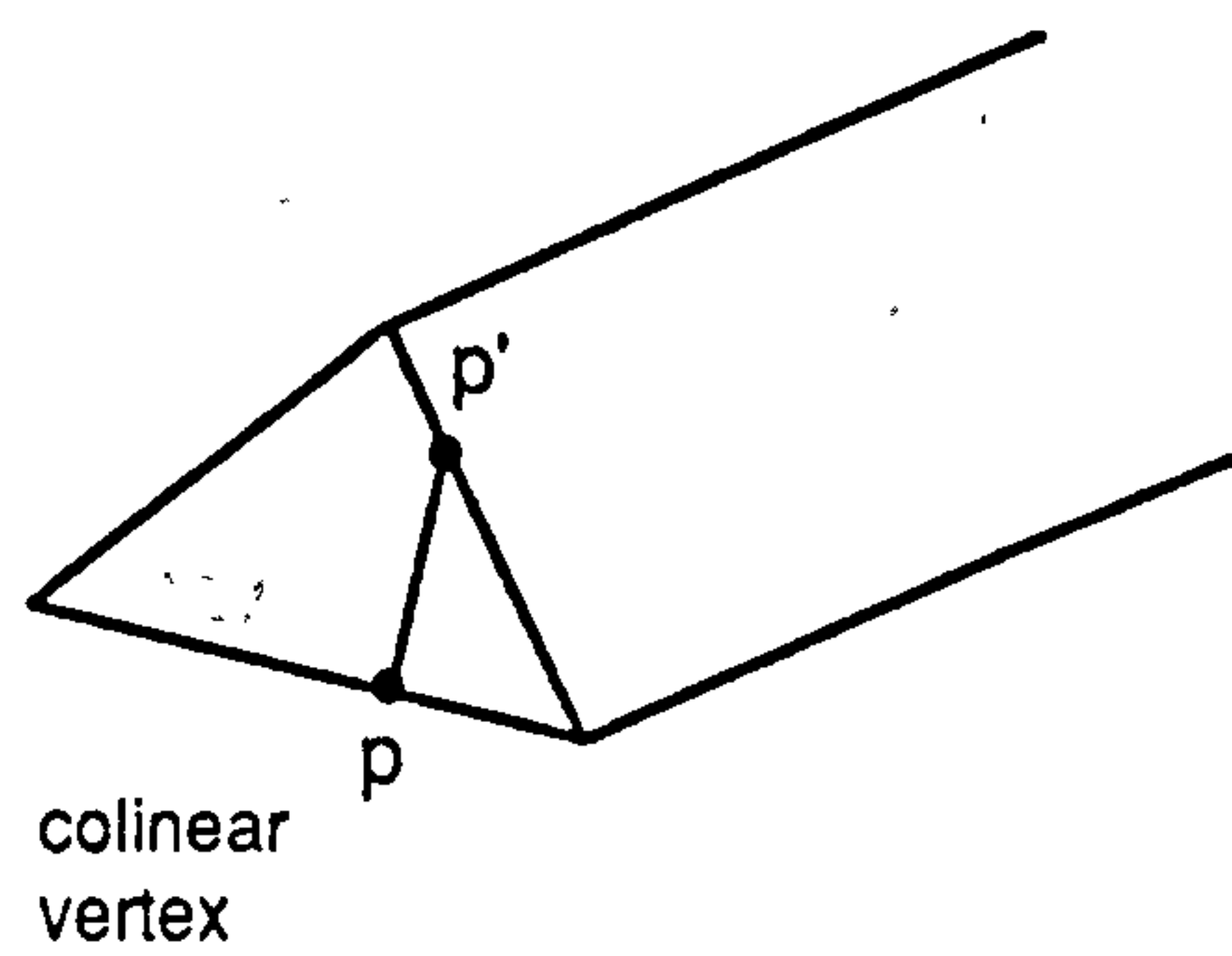
c) A planar ceiling face must be created. As it is possible for the swept co-ordinates of some of the 'corner' vertices to be coincident, it is necessary to find 3 non-coincident and non-linear vertices to define the plane, and then check that all remaining vertices lie on the plane. It is possible that all vertices are coincident, i.e. all side faces intersect in a point, or that only 2 such vertices are found, i.e. all side faces intersect in a line. Again, in these cases the topology is not modified, a ceiling face exists with zero area, and is flagged as such.

### 10.3.2 Creation of Internal and External Wall Constructions

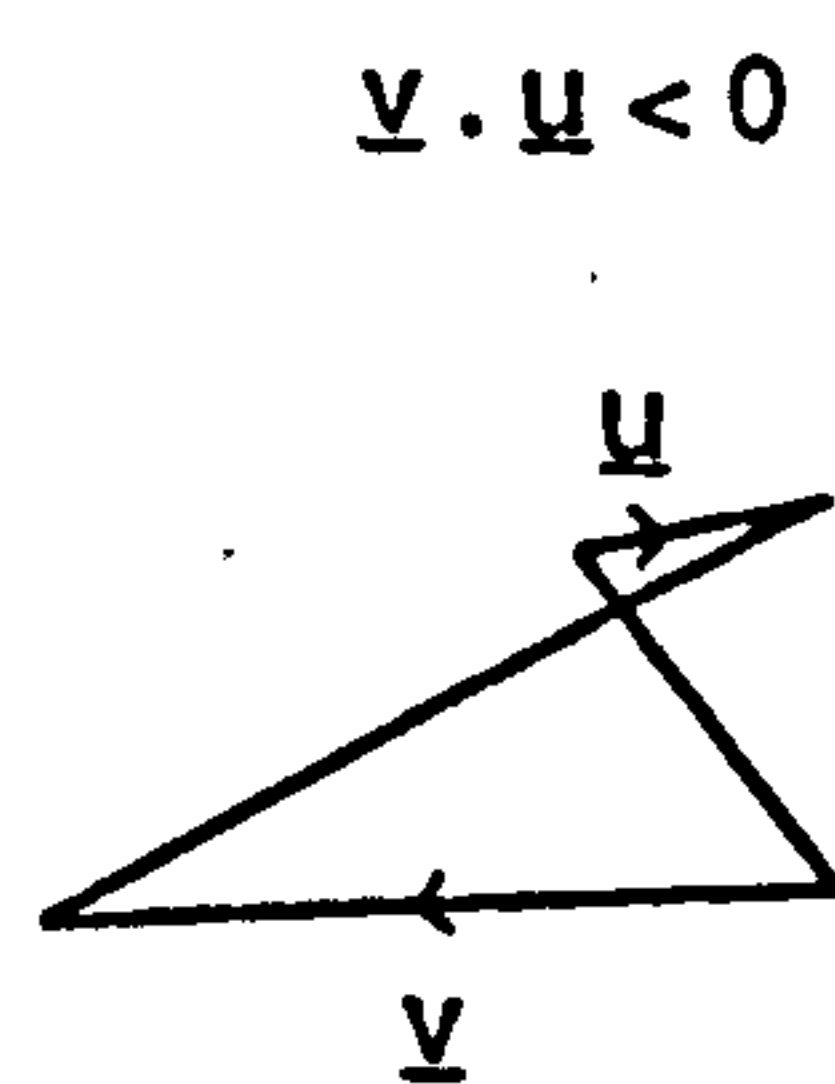
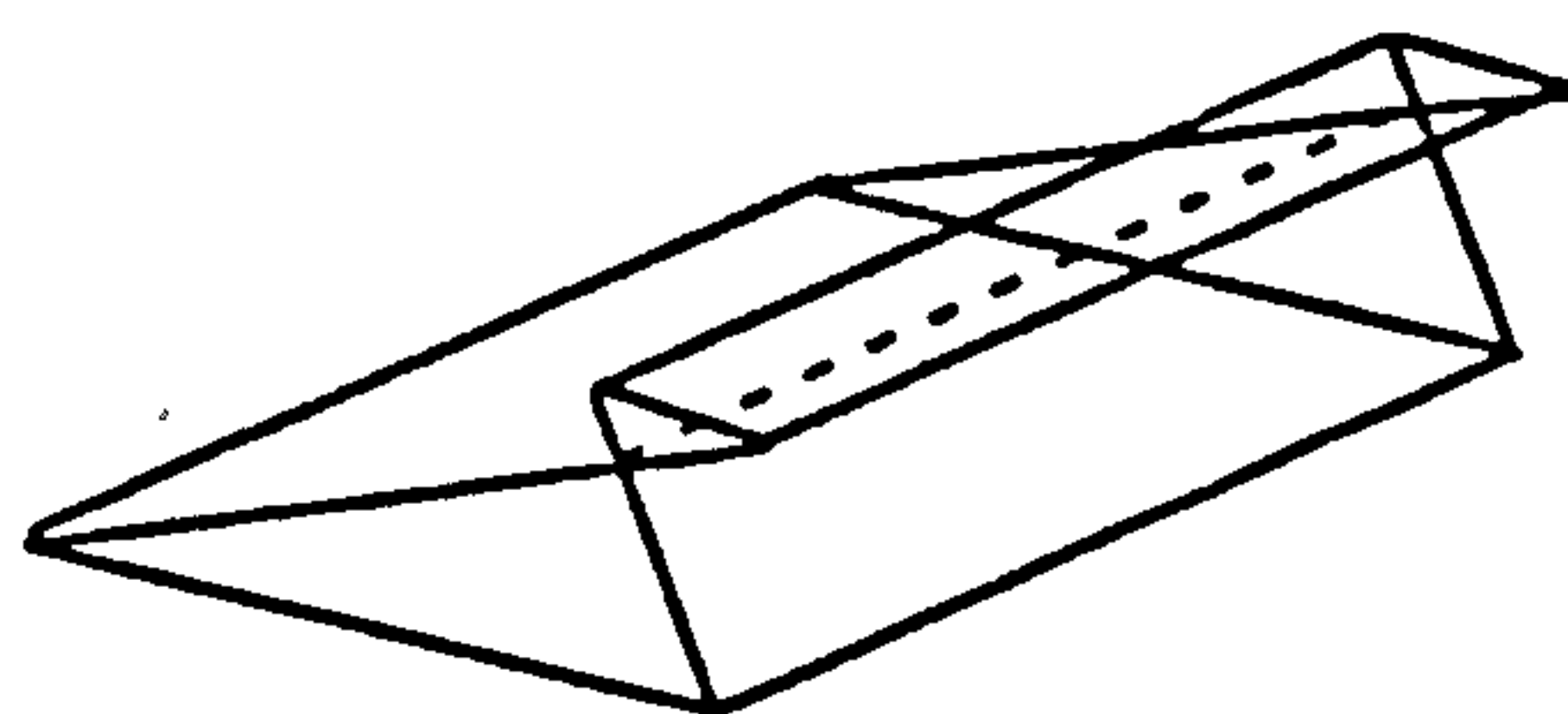
The process to create the internal and external constructions resulting from sweeping one graph edge is a simplified version of that described in 10.1 to create floor/ceiling constructions. It is simplified in the following way:

a) Only two space faces need to be compared, they are known to overlap as they share a common edge, corresponding to the original graph edge.





**Figure 10-20** Intersection of sweep vector of colinear vertex with 'up' edge of corner vertex



**Figure 10-21** Check for well-formed side faces

b) Both faces are simply connected and convex, they each have an outer loop of four edges.

c) Only one 'overlapping' face can be defined, whose edges must be 'added' to both existing wall faces, so defining the internal wall construction. An external wall construction is defined by each of the remaining faces.

### 10.3.3 Deletion of Wall Construction to merge spaces

As discussed previously, a complex 3D space can be modelled by two or more spaces that are merged together after the sweep operation, by deleting the wall constructions between the spaces. The originating graph edge needs to be flagged to indicate that a special 'invisible' wall construction is created. The resulting side faces created from the graph edge must be geometrically identical, i.e. only one internal wall construction created.

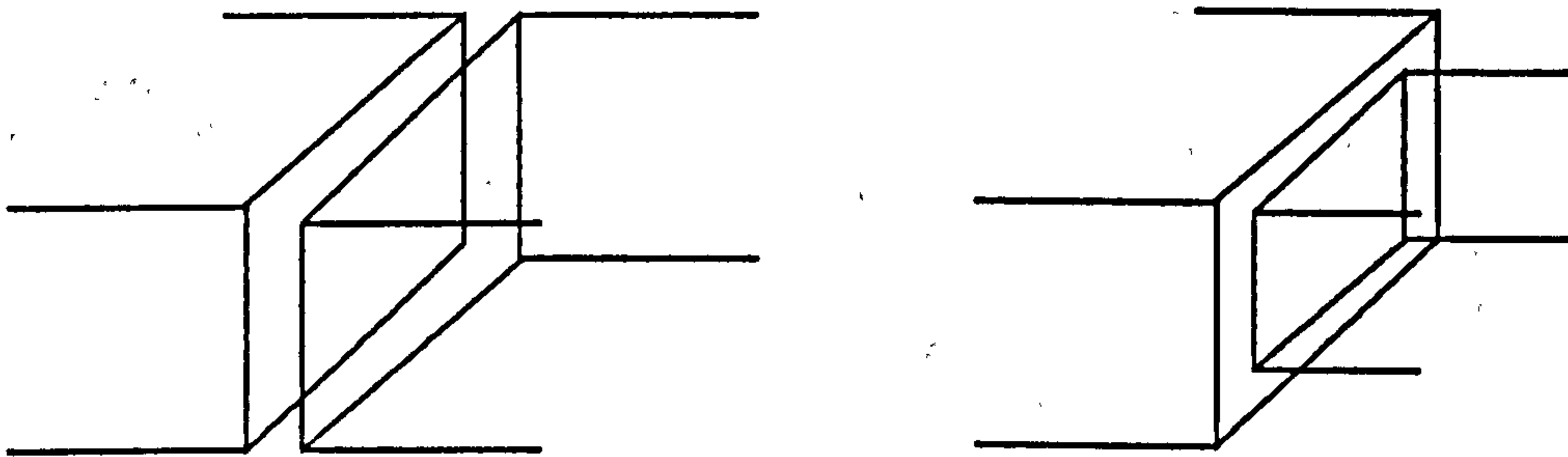
The required 'Delete construction and space' operation can then be derived by using a standard Euler operation: 'Make a hole and kill a body and face' to merge the two spaces. The steps are therefore:

- i) Delete wall construction referencing both faces.
- ii) 'Make a hole and kill a body and face' to merge the winged-edge representations of the two spaces as shown in figure 10.22a), so deleting the first side face.
- iii) Add edges as shown in figure 10-22b) , so creating 3 faces.
- iv) Delete edges as in figure 10-22c), so deleting 4 faces, including the second side face.
- v) As vertices are co-incident, merge edges, deleting vertices, as in figure 10-22d), and also corresponding node instance data from node list.

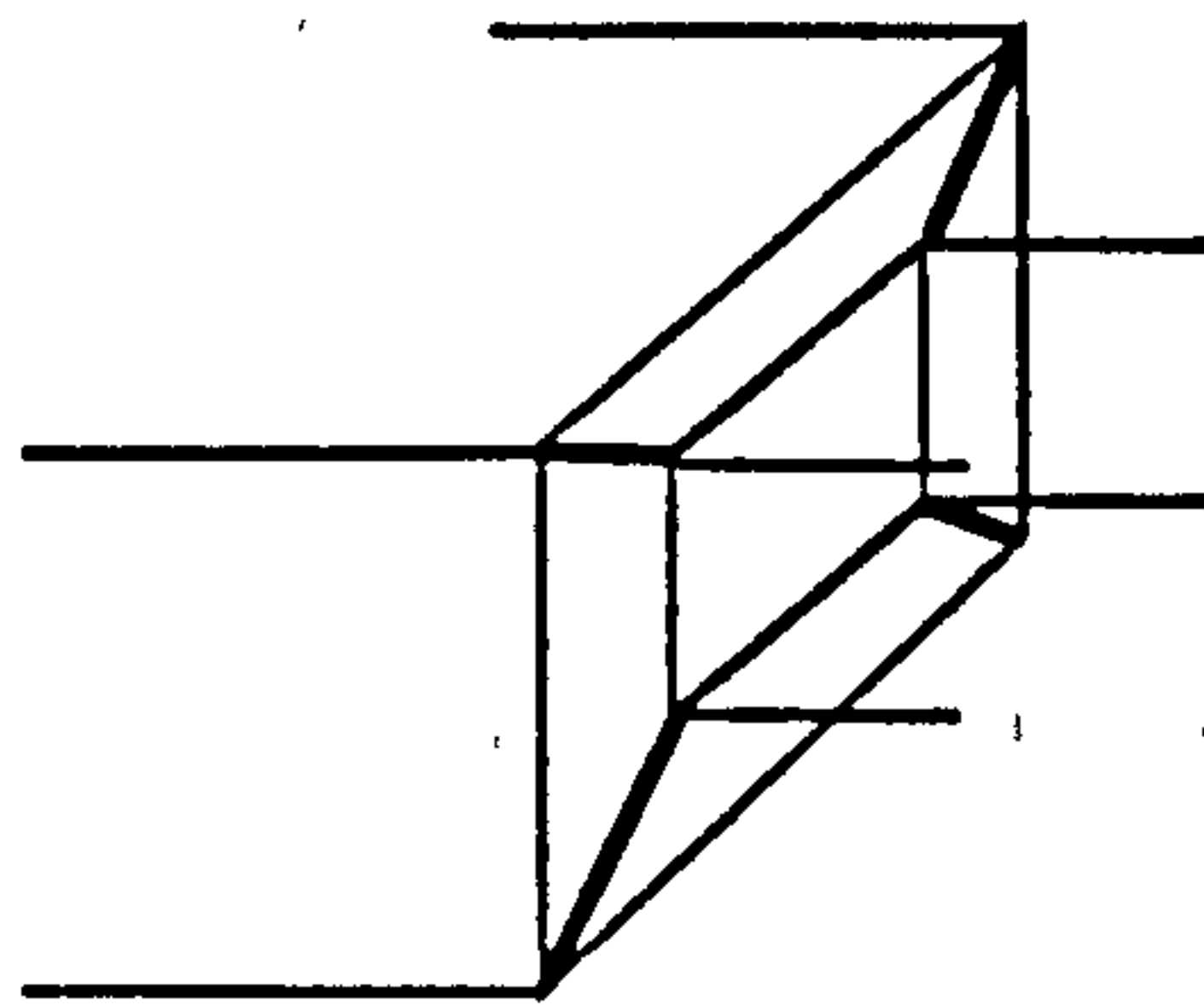
### 10.3.4 Creation of Extended Spaces

As discussed previously, external wall constructions would be created to represent the extension of a space to an upper floor. The following must be implemented to allow these external wall constructions to become internal wall constructions, when adjacent spaces are added on the upper floor:

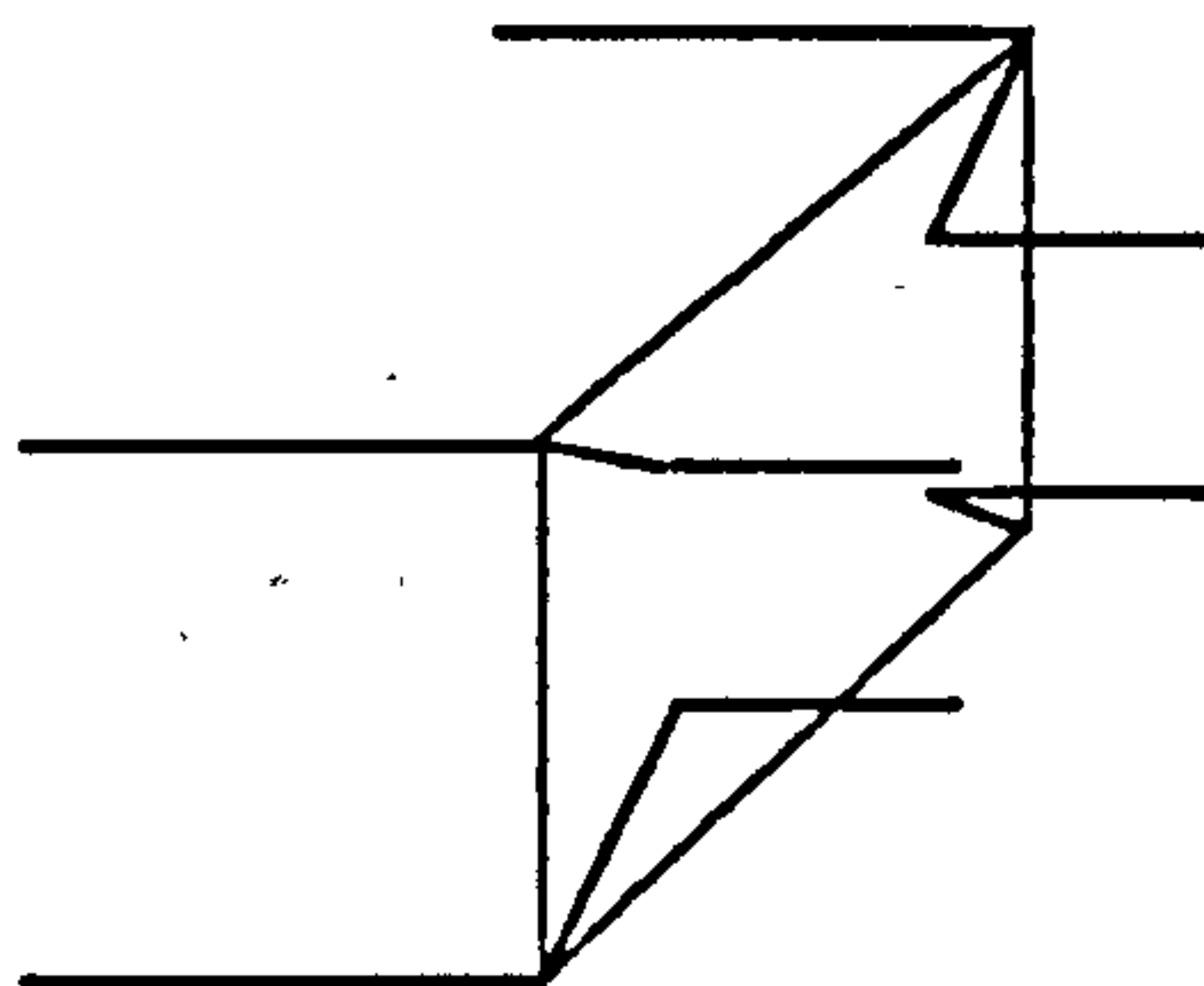
- a) the building model must be interrogated in such a way, that the edges of a planar graph can be created to represent these constructions, and such that this graph then becomes the starting graph of the upper floor.



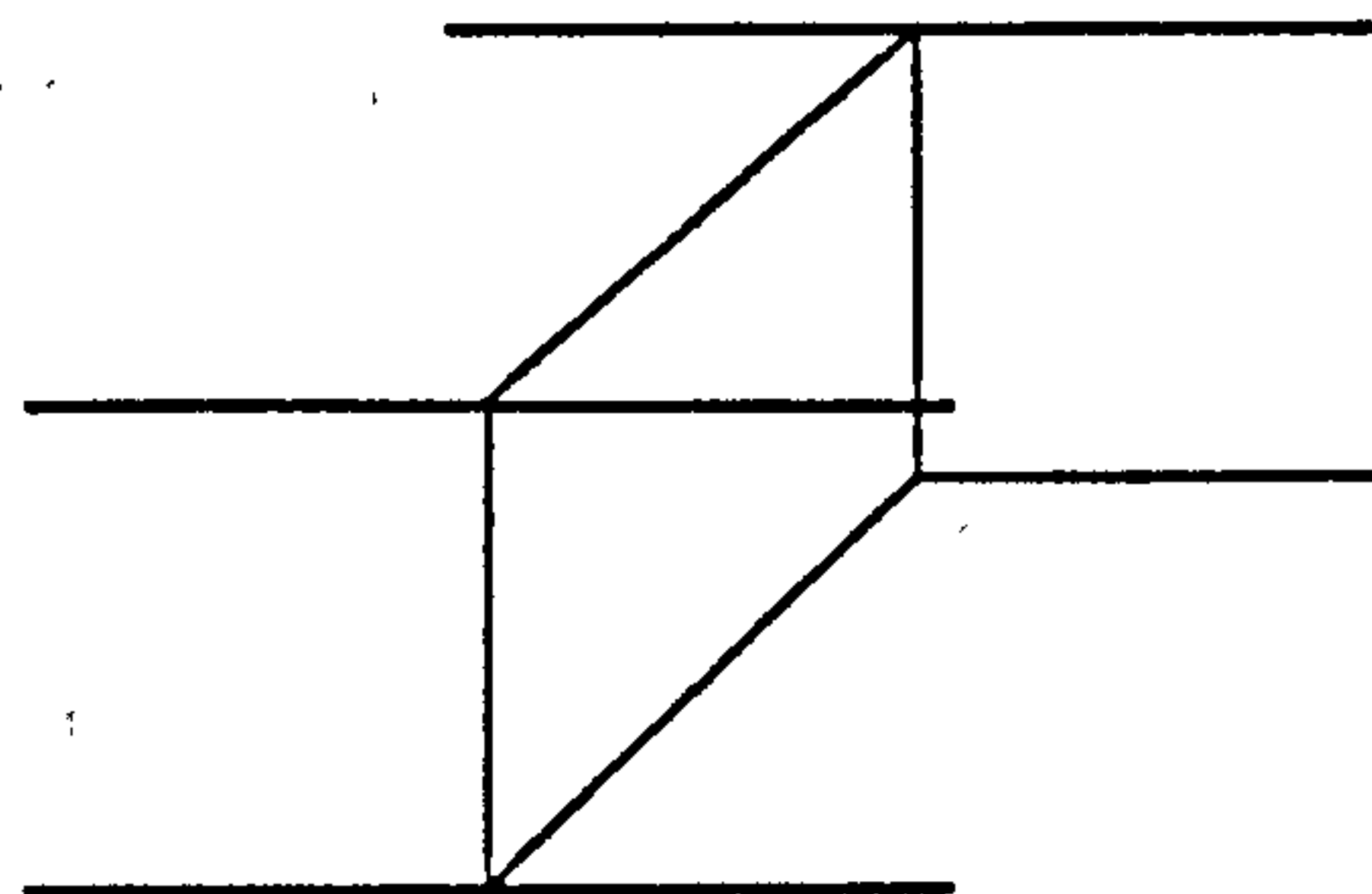
a) Make a hole, kill a body and face.



b) Add edges and create new faces.



c) Delete edges and faces.



d) Merge edges.

Figure 10-22 Process for 'Delete Construction and Space' operation

b) The sweep operation that creates the 3D model requires modification such that those faces representing the 'holes' into the lower floor(s) are not swept, and the existing nodes and wall constructions are used. However, it is possible that modification of these constructions is necessary. For example if a wall is added, splitting a graph edge, such that it is necessary to divide the construction vertically, see figure 10-23. Similarly, if the geometry i.e. height, inclination of the construction needs to be modified.

The above points can be considered in conjunction with the broader requirement for a method of editing/modifying a floor, to allow for the addition of new walls, or the deletion of existing walls. In this case the graph of the entire floor needs to be re-generated, and existing nodes and constructions may need to be deleted.

This requirement can be simplified by the following:

a) The sweep operation always creates space units of the same basic topology: a horizontal 'floor' face and a 'ceiling' face, with side faces each corresponding to the originating graph edge. New nodes and constructions are always created.

b) A graph of a floor can be generated by assuming that all of the space units of a floor can be identified, and therefore the floor face of these can be accessed. The edges of these faces can then be used to generate the floor graph. The other geometric data: vertex heights and inclinations is then interpreted from the wall constructions adjacent to these floor face edges.

With the above implementation, a space extending over more than one floor can be represented by a set of vertical space 'units', with additional data in the building model to link together the units into a set. The sweep operation is then modified to create a space unit for each floor within the range of the space. This assumes that the vertical positions of all upper floors are known in advance. See figure 10-24.

The above brings into question, how much 'evaluation' of the model there should be while a building model is still being input and modified. As indicated above, modification is simplified if the basic topology of a space unit is always maintained.

A fully evaluated model is required for analysis, when areas of constructions and volumes of spaces must be calculated. Then and only then is it necessary that the following processes so far described are complete:

- a) Generation of floor/ceiling constructions between floors
- b) Division of wall faces into internal and external construction faces.
- c) Merging of adjacent spaces.

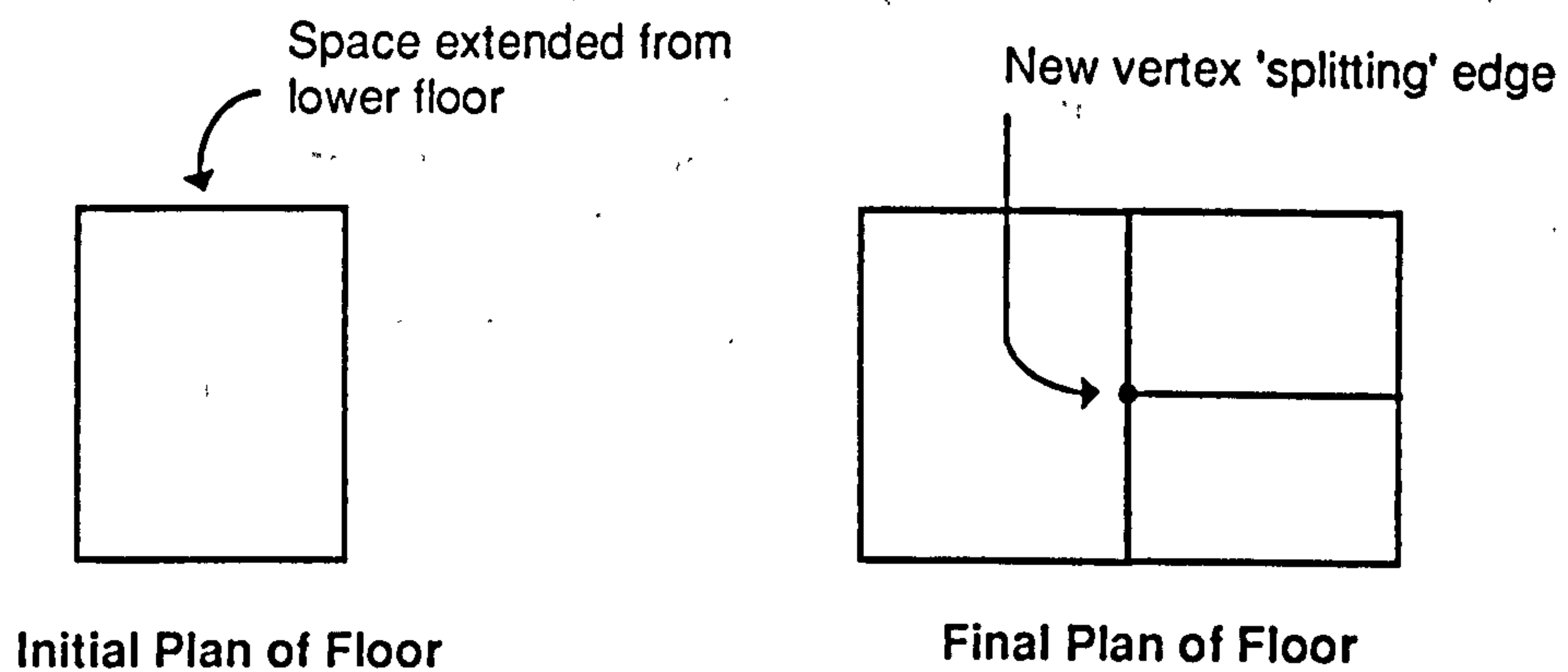


Figure 10-23 'Split edge' operation, implying modification to original wall construction

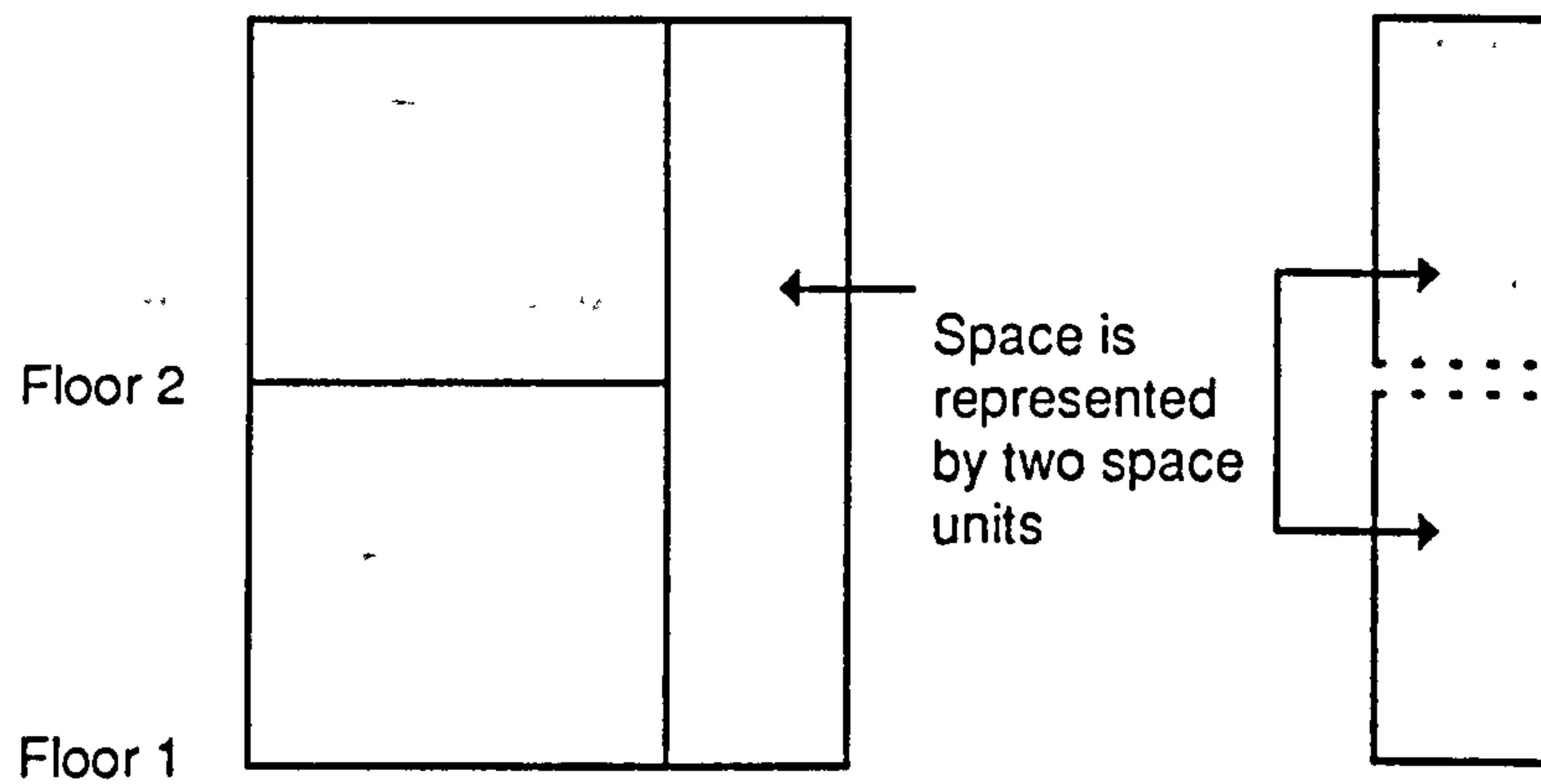


Figure 10-24 Representation of space extending over two floors by two space units



While the building model is still being created only partial evaluation is necessary.

The sweep process to create the 3D space units from the 2D graph is the minimum necessary evaluation in order that the full geometric data is used to generate any upper floor space units for inclusion into the planar graph representation of an upper floor. It is also required to generate 3D displays of the model while it is being created to aid the user.

Until a full evaluation is required, a space can be represented as a set of basic topological space units, linked horizontally by special wall constructions, and vertically by special floor/ceiling constructions. In this un-evaluated state, each wall construction corresponds to one graph edge, but may refer to two non-identical faces i.e. they are not 'proper' wall constructions.

### 10.3.5 Representation of Space as set of Units

The above method to create extended spaces implies modifications to the data structures representing one space. It is necessary to be able to determine the following for each space:

- a) the vertical floor range of the space
- b) the set of space units representing a space, i.e. a space unit corresponding to each floor in the above range
- c) the 'floor' face of each space unit such that the basic topology of the unit can be assumed using this face i.e. a wall construction is adjacent to each edge of this face.
- d) the ceiling and floor faces that are co-incident acting as the link between the space units as the nodes between the floors will not be merged.

The data record for a space is therefore extended and a new data item, a 'spacelink', is introduced. These are described in tables 10-1 and 10-2 and illustrated in figure 10-25.

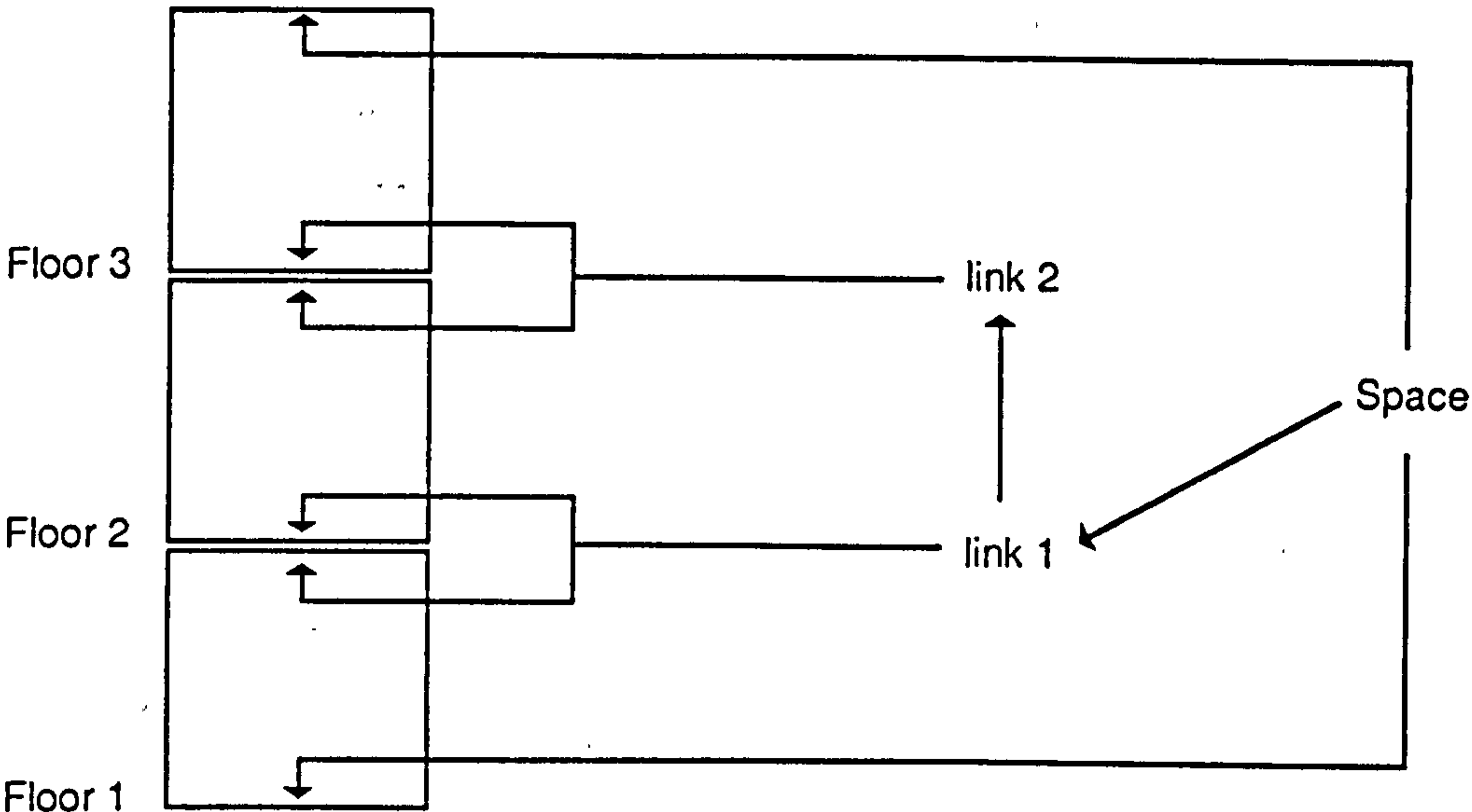


Figure 10-25 Representation of space as set of units

**SPACE**

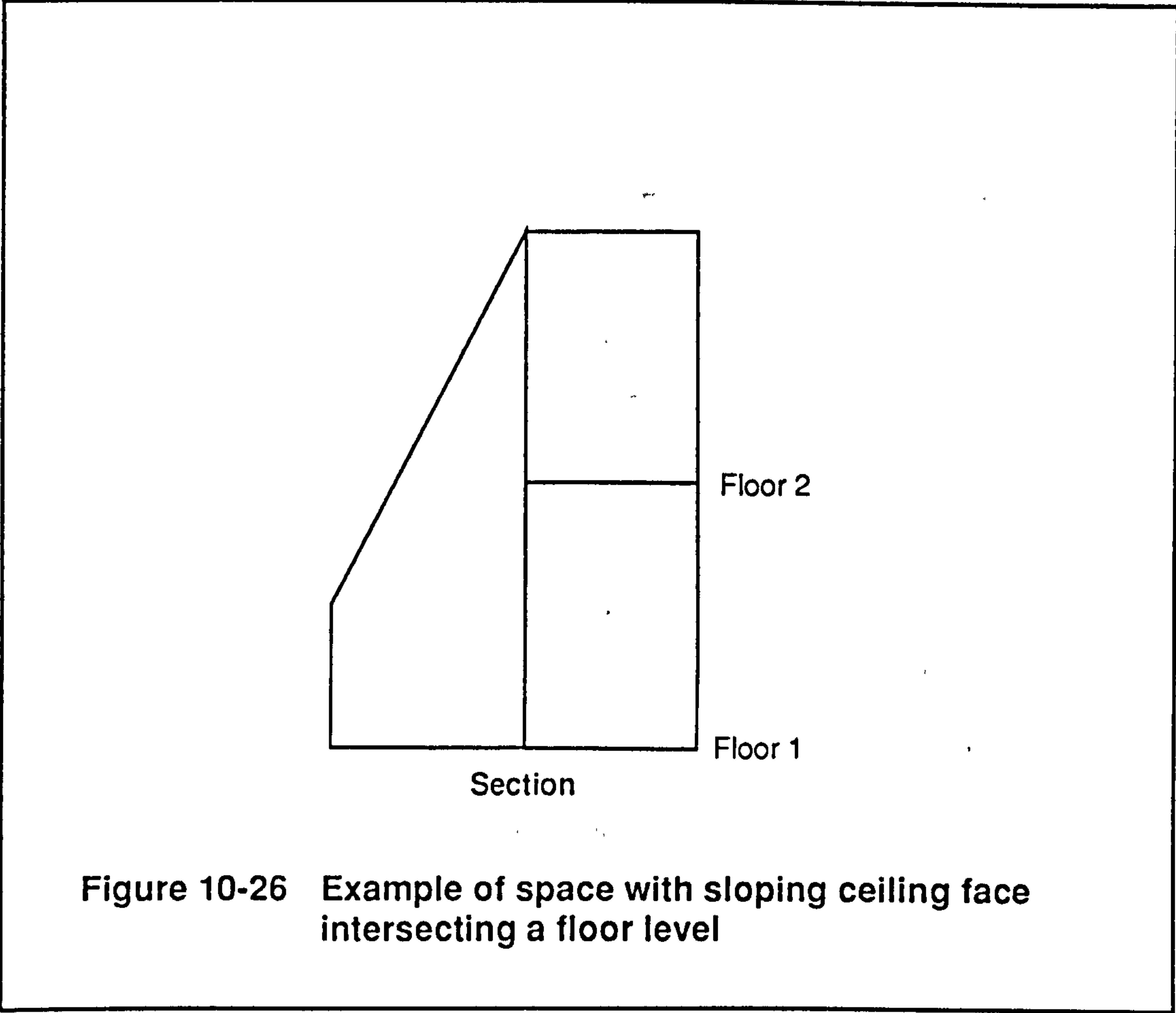
Field Name	Description
nextspace	Next space in list for building
lowerfloor	No. of lowest floor of range of space
upperfloor	No. of highest floor of range of space
floorface	Floor face of space unit of lowest floor
ceilingface	Ceiling face of space unit of highest floor
spacelink	Space link, null if range is one floor only

Table 10-1: Data Record of Space represented as set of units**SPACELINK**

Field Name	Description
space	Next construction in list for building
floorface	Floor face of space unit
ceilingface	Ceiling face of lower space unit, co-incident with floorface
spacelink	Space unit, null if last spacelink in floor range

Table 10-2: Spacelink Data Record**10.3.6 Intersection of Ceiling Face with Floor level**

The above representation, simplifies the case where a sloping ceiling face is defined that intersects a floor level, i.e. the space extends to an upper floor in some side faces but not all. An example is shown in figure 10-26. By creating an 'invisible' dividing wall construction that will later be deleted to merge the horizontal space units, the space can be specified as a set of units as shown in figure 10-27.



**Figure 10-26** Example of space with sloping ceiling face intersecting a floor level

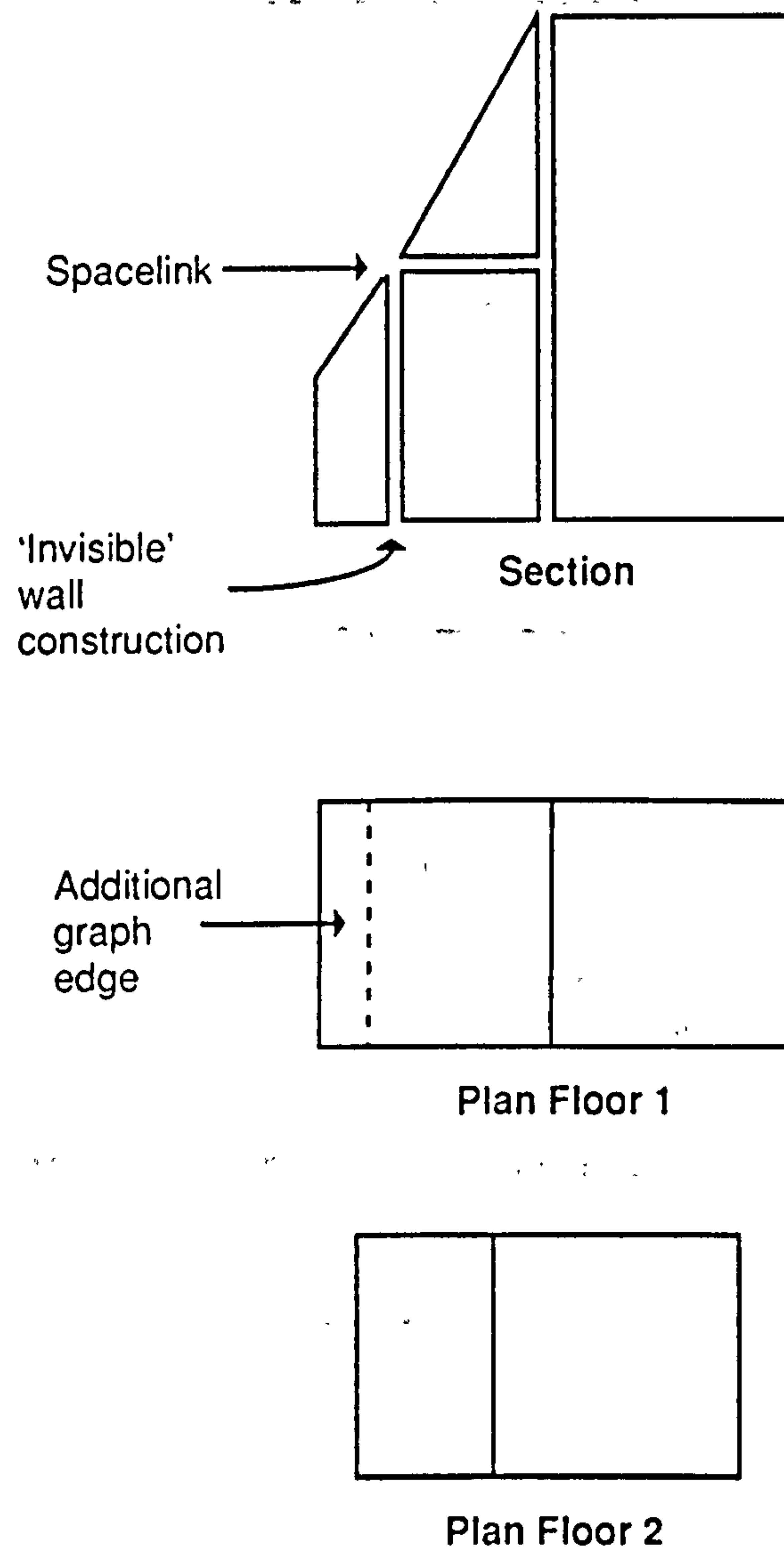


Figure 10-27 Division of space into space units



The positions of the 'invisible' wall construction(s) are defined by the intersecting line of the plane of the sloping ceiling face and the horizontal plane at the given floor level. As the ceiling face may be concave, there may be more than one construction that must be defined. The intersection points of this line with the edges of the ceiling face are calculated, and ordered along the line. The (x,y) co-ordinates of each pair of points then defines the endpoints of the required graph edges, from which the invisible wall constructions can be created, the z co-ordinate defines the sweep height of each vertex.

The space unit of the upper floor does not require a side face. To maintain a standard topology a side face is created with zero height. It then becomes acceptable for any side face to be specified by a user as having a zero height.

### 10.3.7 Modified Sweep Operation

To include the requirement to create extended spaces in the representation described above, and using a sweep direction and height for each vertex, the sweep operation described as having two main steps in Chapter 9 is modified into the following steps where steps 2 and 3 correspond to the original steps 1 and 2, except they are repeated for each floor. Those graph faces identified as being 'external' to represent courtyards are detected and omitted from the process below.

1. Calculation of the geometry of each space, as specified in 10.3.1: from this the uppermost floor reached by all the spaces starting at the current floor is found. If there are any sloping ceiling faces that intersect a floor level, additional graph edges are then added to represent this intersection, before proceeding with the creation of the space units.

For each floor from the current to the uppermost:

2. Calculation of co-ordinates of the intermediate points at the floor level and at the next floor level, the latter then becoming the swept point co-ordinates. This uses the geometry calculated at step 1. Where the point  $\underline{p}$  is the point to be swept and  $\underline{d}$  is the direction of sweep, then the co-ordinates of the point  $\underline{p}'$  at an intermediate floor level, which is at height  $h$  above the lowest floor level is given by:

$$\underline{p}' = \underline{p} + \mu \underline{d}$$

where  $\underline{d}$  is the normalised vector  $l_i + m_j + n_k$

and  $\mu = h/n$

The upper and lower nodes are then created.

3. Sweep operation to create the winged-edge data structure of each space unit using the 'corner' vertices only, followed by the addition of further 'up' edges for the colinear vertices. The nodes created in step 2 are referenced, and the wall constructions are created. If a unit is not the highest unit of the

space, a new space link entity is created and references the ceiling face of the unit, otherwise the space entity refers to the ceiling face. If a unit is not the lowest unit of a space, then the existing space link created previously references the floor face, otherwise the space entity refers to the floor face.

### 10.3.8 Creation of Planar Graph from Building Model

Given the extensions to the building representation described above, it is possible to determine all the spaces that are relevant for a given floor, and hence the floor face of the appropriate space unit of each space. These 'floor' faces are then used to generate the planar graph.

The process is a simplified version of the one described in Chapter 8, such that each edge of the 'floor' face is 'copied' to create the planar graph. The process is simplified by the following:

- a) For each endpoint of a new line segment, corresponding to a vertex of the originating edge, either a corresponding graph vertex exists or it is within a face. It cannot be 'on' an edge.
- b) The edges are added in sequence as determined by the loop of the floor face, and therefore the first vertex of a line segment is always the last of the previous edge. (Except for the first of the loop).
- c) For each line segment, corresponding to the originating edge, a corresponding graph edge may exist, previously created by processing the floor face of the adjoining space unit. Otherwise, there is no need to check for intersections with other edges.

It is necessary to be able to identify the originating space for each graph face, such that the subsequent sweep process, after new graph edges have been added etc., can insert the space unit into the set relevant to the space. This will also identify those graph faces which are 'external' i.e. have not originated from a space, and which therefore do not need to be included in the subsequent sweep process.

When a graph face is created, from an 'Add edge and face' operation, such that the edge is a copy of the last edge of the outer loop of the floor face, then this face is indicated as corresponding to the space. It is possible that a graph edge already exists corresponding to this last edge, in which case it is the last face created by an edge copied from this outer loop, or if no face is created, i.e. all edges exist already, the appropriate face is determined.

All other faces, as created from inner loops are indicated as 'external'. These faces may not remain 'external' at the end of the process, as they may be 'filled up' with one or more faces corresponding to floor faces processed subsequently.

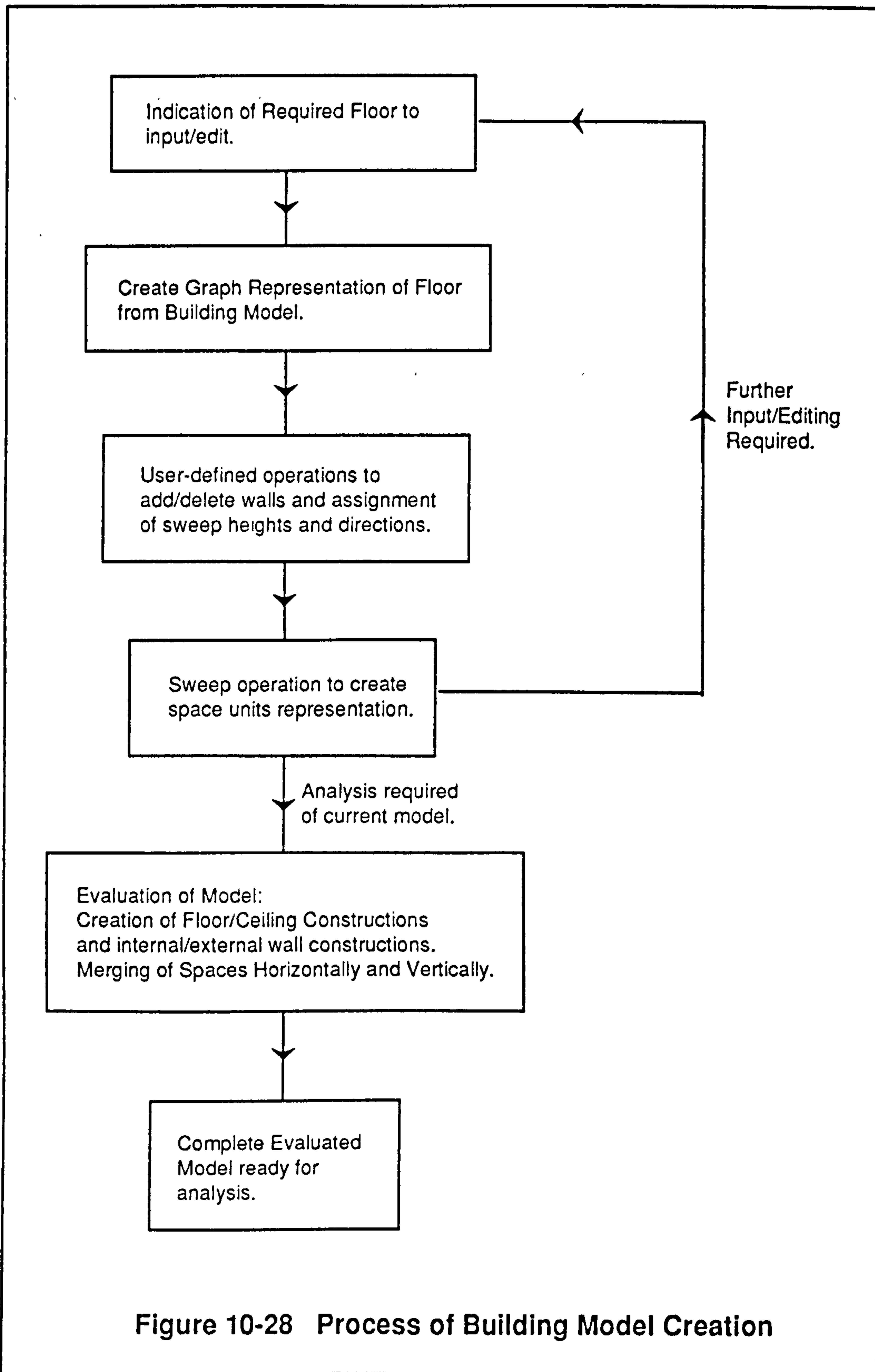
#### 10.4 Summary of Building Model Creation

This chapter has discussed the means by which more complex building modelling can be achieved, and to allow the input and subsequent editing of a floor definition. For clarification this process is summarised in figure 10-28.

There is a restriction attached to this method in terms of the order in which the model is created, in that the model must be created from the lowest floor upwards. A space extending to an upper floor must be defined in the graph of its lowest floor before the upper floor graph can be input/edited. Once an upper floor graph has been added to, and spaces exist which start at that floor, if spaces are added to a lower floor which extend upwards, it must be ensured that the space units created do not intersect with or are adjacent to those already created on the upper floor.

Subsequent editing of such spaces is also more restrictive than of spaces which span only one floor, in that a wall placed on a lower floor extending to upper floors can not be deleted while there are spaces adjacent on the upper floors, i.e. the wall construction is internal on these upper floors.

The completed building representation is the fully evaluated model, such that any analysis is then possible. So that analysis can be performed and then the building model edited further, a copy of the original unevaluated model must be retained. This allows for the required iterative process, such that an initial building model may be created and analysed, and subsequently more detail added as the design progresses.





## 11. ASSIGNMENT OF MODEL ATTRIBUTES

To enable the analysis of the building model, attributes must be assigned to the model defining:

- a) the type of all constructions
- b) the zone to which a space is allocated.

The method by which this assignment is achieved and the required extensions to the building data structures are now discussed.

### 11.1 Allocation of Construction Type

The construction type, that is the layers of materials, must be defined for every construction in the building model. As described in Chapter 3, a table of Building Elements can define the set of construction types, from the construction database, that are used to specify the surfaces of a building. This method is applicable also to the building model representation so far described. The names of the Building Elements e.g. internal partition, cavity wall, etc. define the categories of the constructions. It is this category, or Building Element name that is the attribute of the construction.

It is possible that where the analysis requires surface areas, and not detailed geometry, i.e. co-ordinates of vertices, that it may be convenient to allocate more than one category to a construction, instead of defining more constructions by dividing a wall into sections. For example, at the early stages of design, it may only be necessary to study the effect of glazing along a facade, specified in terms of its total area, or as a percentage of the total facade area, rather than positioning the windows themselves. Therefore a second category would need to be allocated to the wall construction.

Before analysis, it is necessary for all constructions to be categorised. This could be a tedious task for a user. This is simplified by the use of default building element names, i.e. each construction is automatically categorised a being one of a pre-defined set of names as shown in table 11-1.

Building Element	Construction Description
External Wall	External Wall Construction i.e. adjacent to one space
Internal Wall	Internal Wall construction i.e. adjacent to two spaces
Ground Floor	External Floor Construction, on lowest floor of the building
Upper Floor	External Floor construction, on an upper floor i.e an overhang
Ceiling	External Ceiling construction on any floor
Upper Floor/Ceiling	Internal Floor/ceiling construction

Table 11-1: Default Building Element Allocation



Obviously, any of these default allocations can be overridden by a user, and any of these default names can be used to reallocate constructions not automatically included in the category. For example, an internal wall construction may require the same construction materials as the standard external wall, in which case it can be reallocated as 'External Wall'. Where a building may be on a sloping ground site, it is possible that an external floor construction on an upper floor may be adjacent to the ground and be of the same construction materials as those floor constructions of the lowest floor. N.B The above allocation of default names does not determine the Building Feature type as described in Chapter 3, this is discussed later.

### 11.1.1 Construction Category Data Entity

A new data entity is introduced for the allocation of categories to the constructions to reference the Building Elements data as shown in table 11-2. Each construction refers to a linked list of these entities, to allow for more than one category. It is necessary to identify the 'main' category of a construction, and this is achieved by setting the percentage to 100, although its true proportion is subsequently calculated to be 100 minus the total proportions set in the other categories. A construction with no reference, a null reference is assumed to be of the default category when the building is analysed.

#### **CATEGORY**

Field Name	Description
buildelem	Reference to Building Element
percent	Percentage of total area of construction
nextcat	Next category for construction

Table 11-2: Category Data Record

### 11.1.2 Assignment of Category to constructions

The assignment is performed by the user when creating the floor graph. This is relatively simple for wall constructions: category entities can be created and temporarily assigned to a graph edge, and then transferred to the resulting wall construction. This becomes complicated when both an internal and external construction(s) are created from one graph edge. At present the assumption is made that these are specified by the same category list.

The assignment to floor/ceiling constructions is complicated in that the constructions are not defined until the evaluation described in 10.1 is complete. Therefore a category is assigned to both the floor and ceiling face of a space, achieved by a temporary assignment to the graph face, and then

transferred to the resulting space. Obviously, this does not apply to 'floor' or 'ceiling' faces referenced by spacelinks.

If all or part of one of these faces is used to create the face for an external construction, then it is assigned the same category as the originating face. Otherwise where an internal construction is created, referencing a floor and a ceiling face, then the assumption is made that in effect the material layers of the floor face are overlaid on top of those of the ceiling face and a composite construction type would be created. Therefore a composite building element name is created, by concatenating the building element name assigned to the floor face by the name assigned to the ceiling face. For example, if a construction is created from a floor face categorised as 'tiled' and a ceiling face categorised as 'suspended', the resulting category of the internal floor/ceiling construction would be referred to as 'tiled/suspended'. This still requires that only one entry from the construction types database is allocated to this name, and does not imply that the material layers within the construction must correspond to those within a type allocated to 'tiled' or 'suspended'. It is naming convention for convenience only, as a means to identify the areas of the floor/ceiling constructions.

Therefore there is requirement for new data items to be added to the wall construction and space entities in the 'unevaluated space unit' representation to reference a linked list of categories, and then in the evaluated floor/ceiling construction entities. The extended space and wall construction entities are shown in tables 11-3 and 11-4.

## SPACE

Field Name	Description
nextspace	Next space in list for building
lowerfloor	No. of lowest floor of range of space
upperfloor	No. of highest floor of range of space
floorface	Floor face of space unit of lowest floor
ceilingface	Ceiling face of space unit of highest floor
spacelink	Space link, null if range is one floor only
floorcat	First of linked list of categories for floor face
ceilingcat	First of linked list of categories for ceiling face

Table 11-3: Space Data Record referencing Categories

### CONSTRUCTION

Field Name	Description
nextcon	Next construction in list for building
lface	'Left' face of construction.
rface	'Right' face of construction.
normal	3 components of normal direction of left face.
catlist	First of linked list of categories for construction

Table 11-4: Construction Data Record referencing categories

#### 11.1.3 Construction Thickness

As discussed in section 6.3, the building model representation described defines all constructions as having zero thickness, this being the requirement of the conceptual model used in analysis. Thickness is considered as an attribute of the construction type, but is not relevant to the topology of the model.

However this conceptual model must be derived from a true model of the building, where obviously the walls, the ceilings and floors do have a thickness. Therefore, ideally a user should be able to define quite precisely the geometry of the building with walls of varying width, and for the modelling system to derive from it automatically the necessary conceptual model.

If this information is held in the model it then becomes feasible to calculate a true internal space volume, as required for the calculation of air exchange, where the width of the walls and the ceiling may have a significant effect. Also the calculation of a construction area will then always be performed consistently when creating the data for an analysis, where the true external and internal areas may differ, sometimes significantly. The 3D display of the building model can also be more realistic.

The construction thickness has therefore been used to calculate further geometry in the building representation, but with no modification to the topology. If there is more than one category entity assigned to a construction, the 'main' category defines the construction thickness. This has been achieved by calculating an offset point for each space vertex, that is in addition to the point of the node of the vertex. These points are calculated by considering the intersection of the faces adjacent to the vertex, but such that the plane of the face is offset from its original position in its



normal direction into the space, where the offset is dependent on the width of the type allocated to the construction of the face.

For a wall construction, the original face position must correspond to either the centre line, or the left or right side of the wall. This position type is determined by the user. The offset position of the 'left' and 'right' faces of the construction can then be determined.

For a floor/ceiling construction, the assumption is made that the original face position corresponds to the floor face, and the ceiling face is offset by the construction width downwards into the space. See figure 11-1.

The calculation of these offset points is included at step 1 of the sweep process, as described in 10.3.7, and is defined as being at the intersection of three planes described below:

a) Corner vertex, floor offset point:

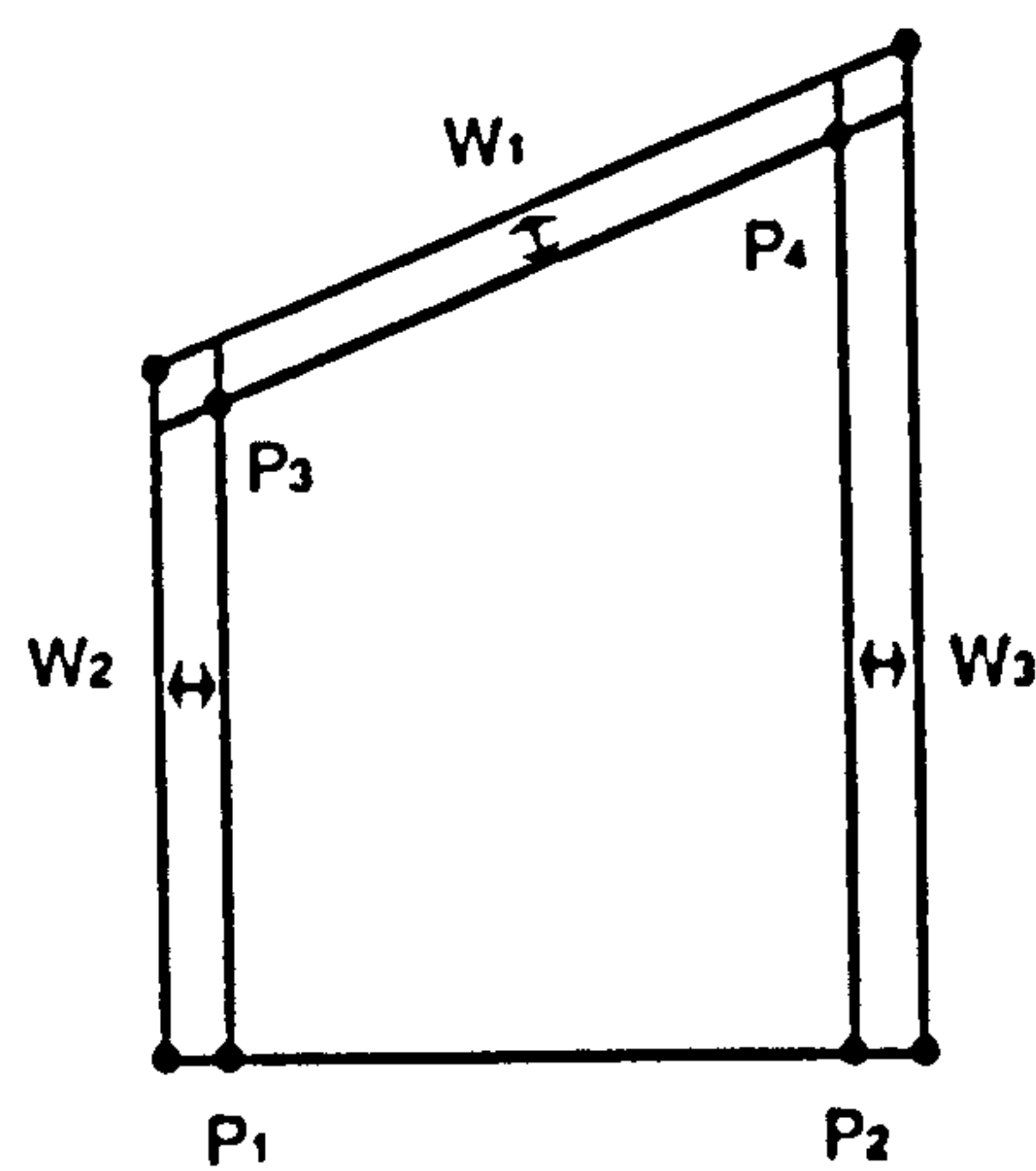
This point is positioned at the intersection of the planes of the offset faces of the two adjoining walls, and the plane of the horizontal floor face. See figure 11-2.

There is a special case when a 'wall' of zero height has been specified, when both the floor offset point and the ceiling offset point are equivalent and calculated to be at the intersection of the offset wall plane, (both walls cannot have zero height), the horizontal floor plane and the offset ceiling plane. See figure 11-3.

b) Corner vertex, ceiling offset point:

This point is positioned at the intersection of the planes of the offset faces of the two adjoining walls, and the plane of the offset ceiling face, calculated from the intersection of the line defined by the sweep vector through the floor offset point and the plane of the offset ceiling face. See figure 11-4.

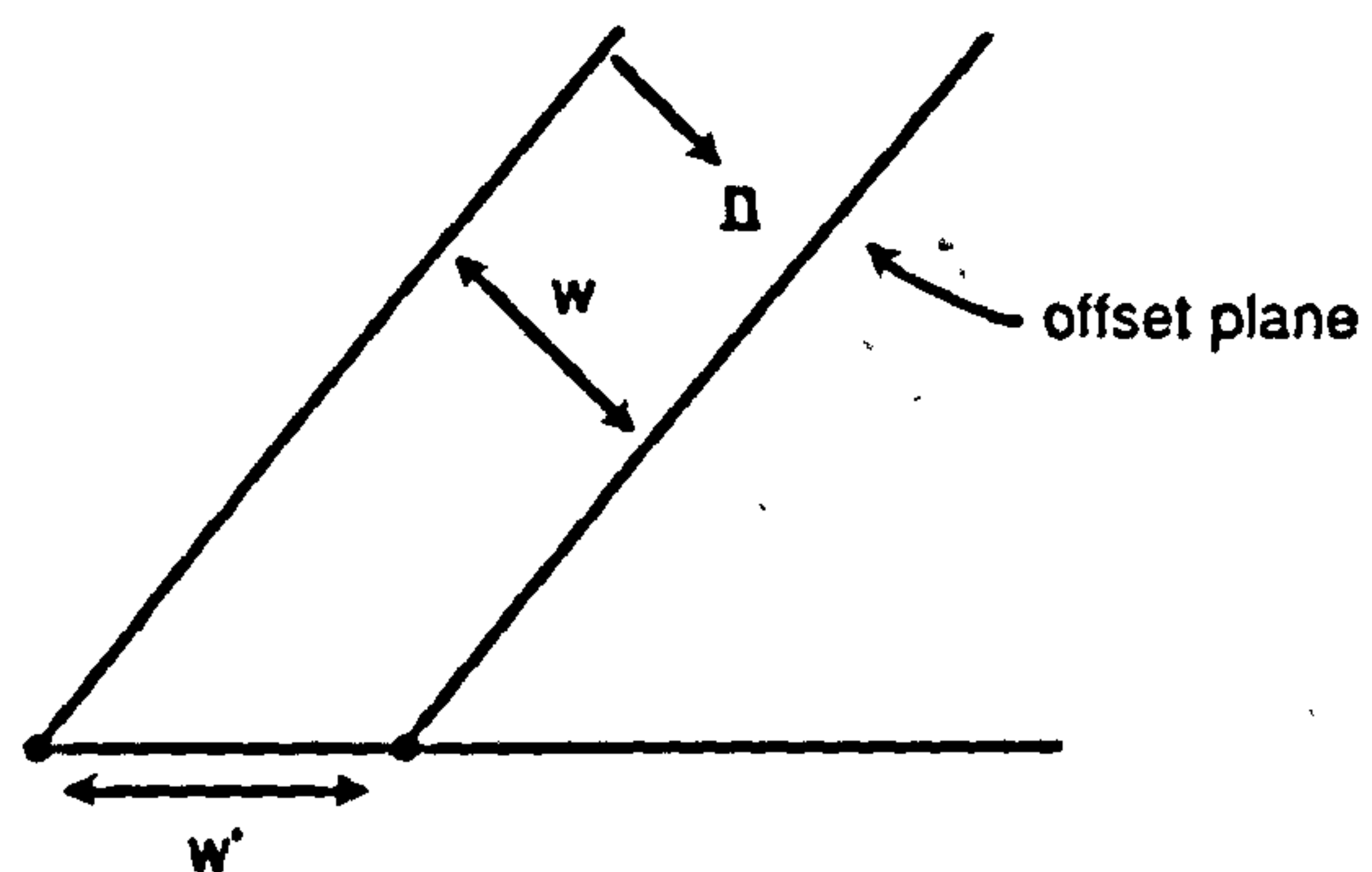
There is a check that this point is not below the floor offset point, i.e. below the floor plane, as is possible with a very low wall, and a thick ceiling construction. In this case, as above, the floor offset point and the ceiling offset point are equivalent and calculated to be at the intersection of the offset ceiling plane and a line through the original floor offset point, with a direction which is the horizontal component of the normal of the ceiling plane. See figure 11-5. This illustrates the most common situation, which is identical to the case of the zero height wall above. However figure 11-6 illustrates another possibility, when the above calculation produces an approximation to the correct geometry, i.e. an arbitrary point is used to represent a line of intersection between the ceiling and floor planes.



$P_1$  to  $P_4$  are offset points  
 $w_1, w_2, w_3$ , are widths of  
ceiling and wall constructions.

**Figure 11-1** Cross-section of space showing offset points at intersections of offset faces





Vertical cross-section of inclined wall,  
normal  $\Omega$ ,  $w$  is width of wall.

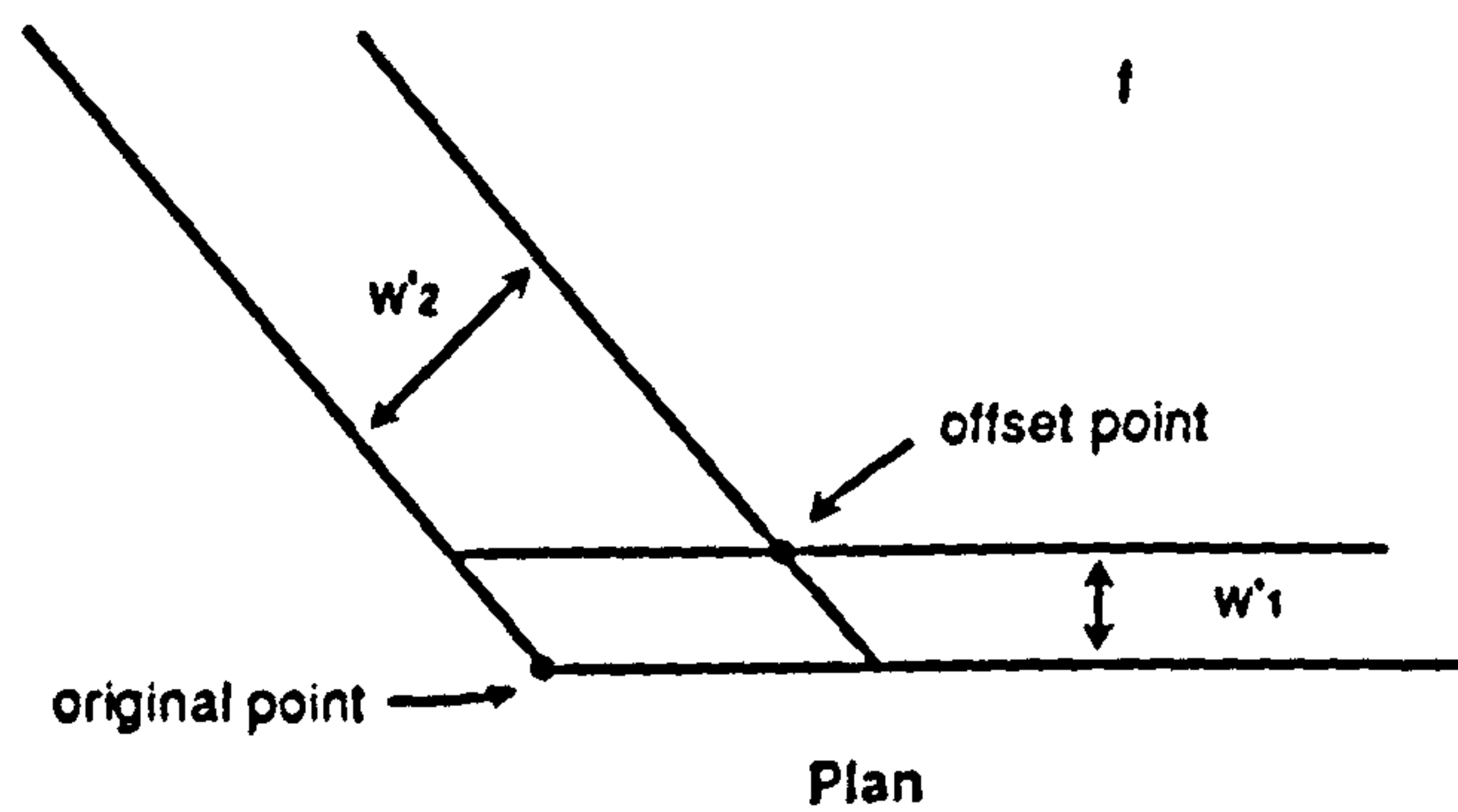
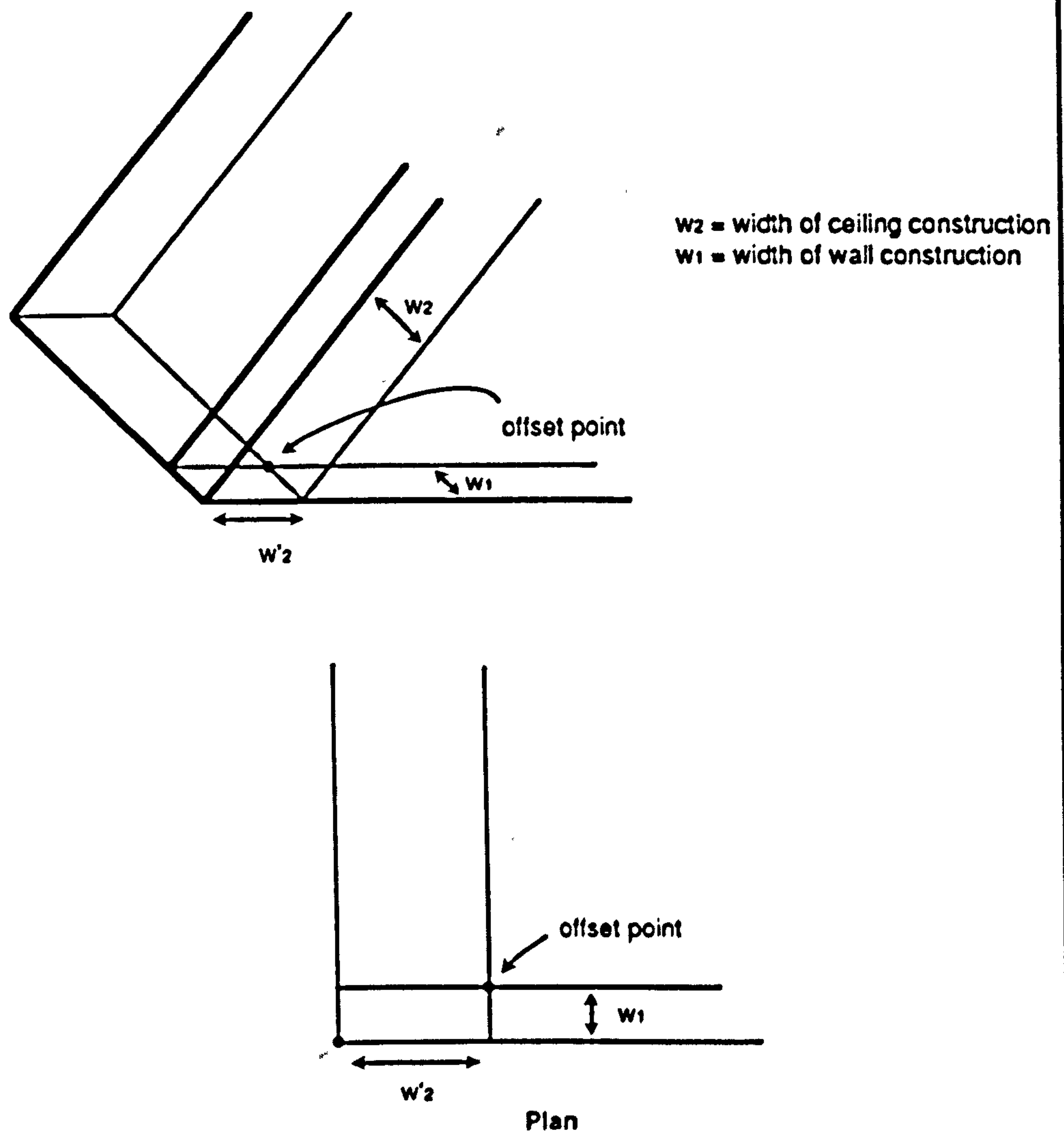
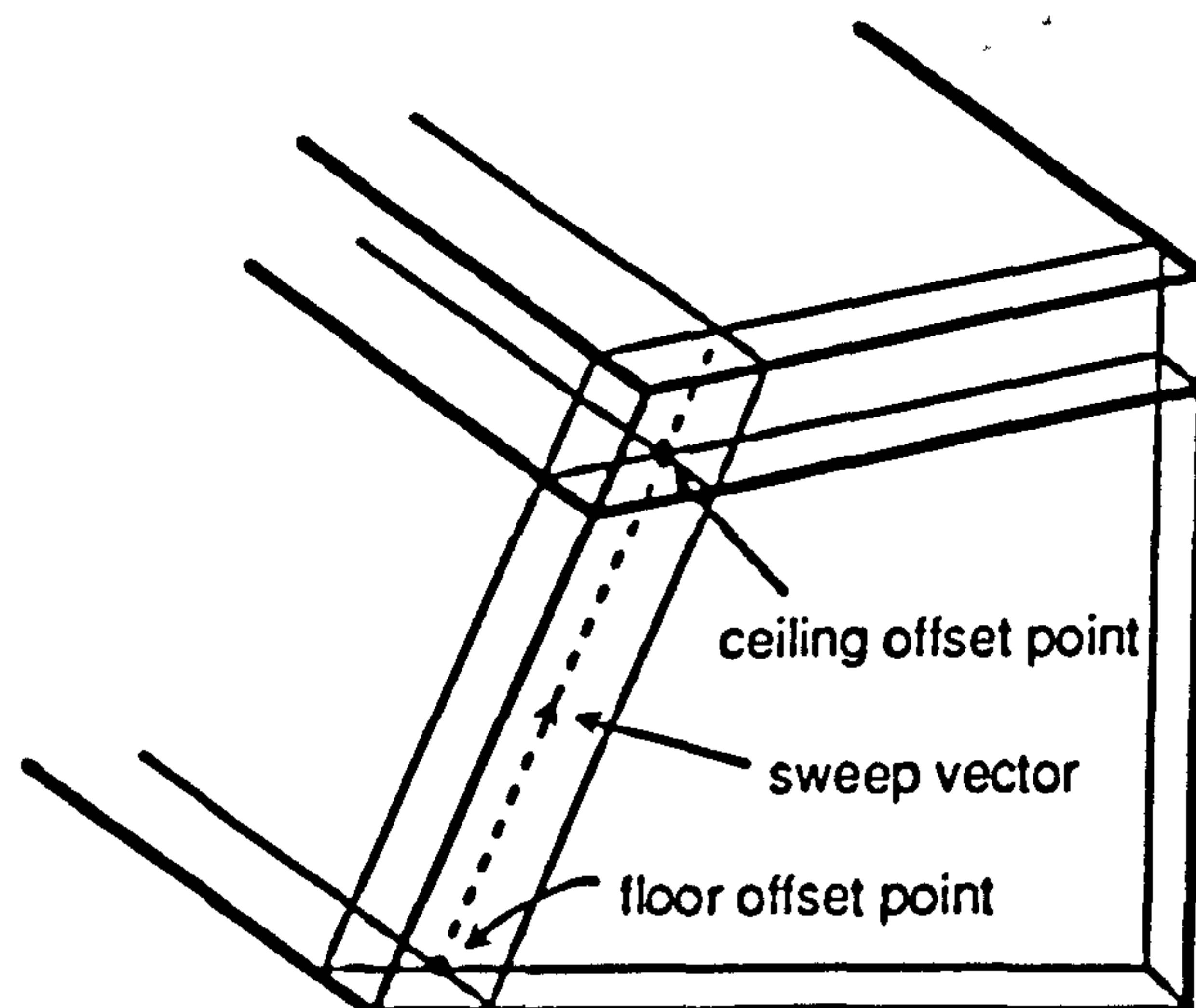


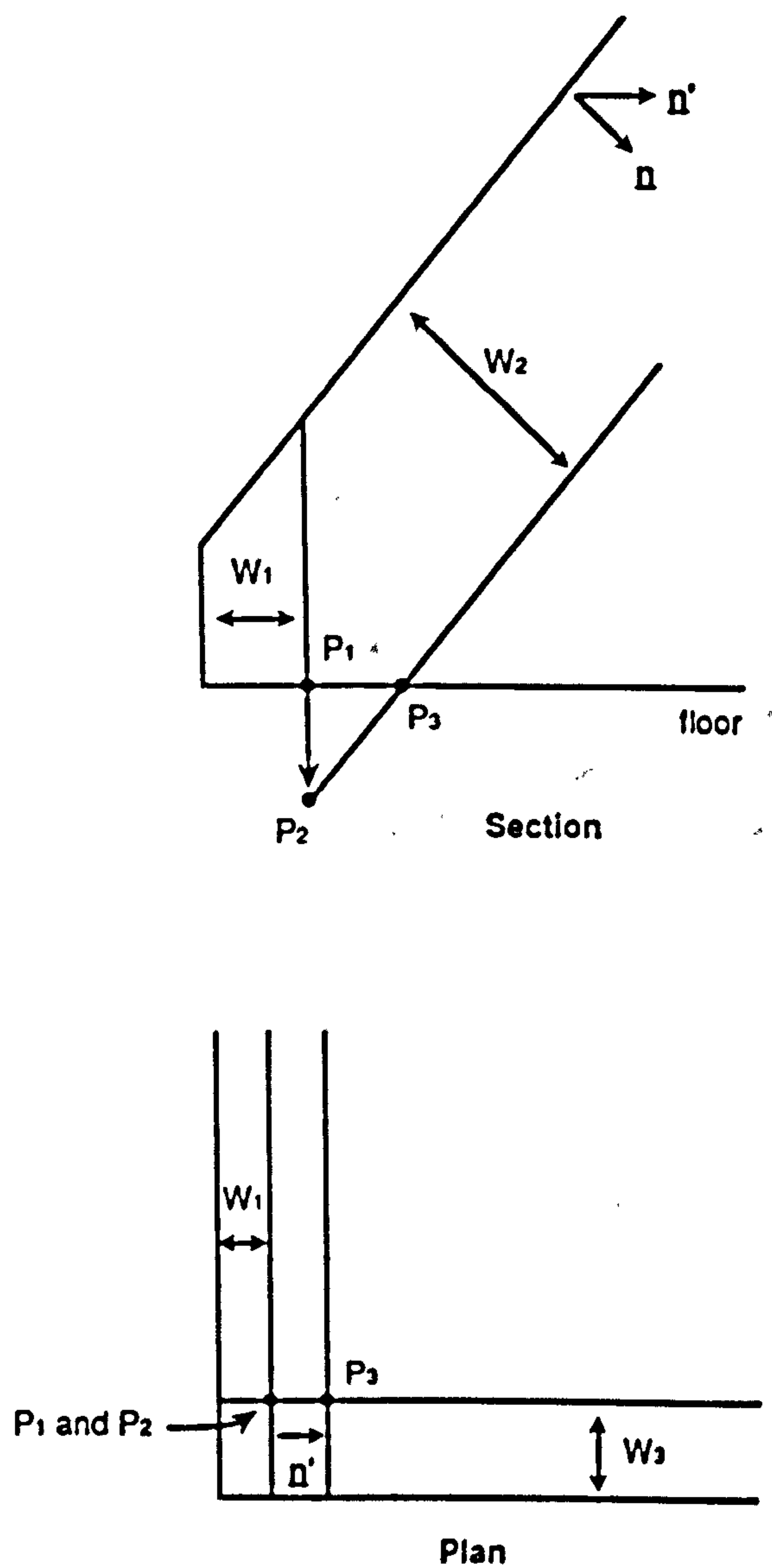
Figure 11-2 Floor offset point at intersection of 2 offset  
wall planes and horizontal floor plane



**Figure 11-3** Floor/ceiling offset points at intersection of offset wall and ceiling planes and horizontal floor plane



**Figure 11-4** Ceiling offset point at intersection of two offset wall planes and offset ceiling plane



$n$  is normal of ceiling plane

$n'$  is horizontal component of  $n$

$P_1$  is floor offset point

$P_2$  is calculated ceiling offset point

$P_3$  is adjusted ceiling offset point

Figure 11-5 Adjustment of ceiling offset point below floor level

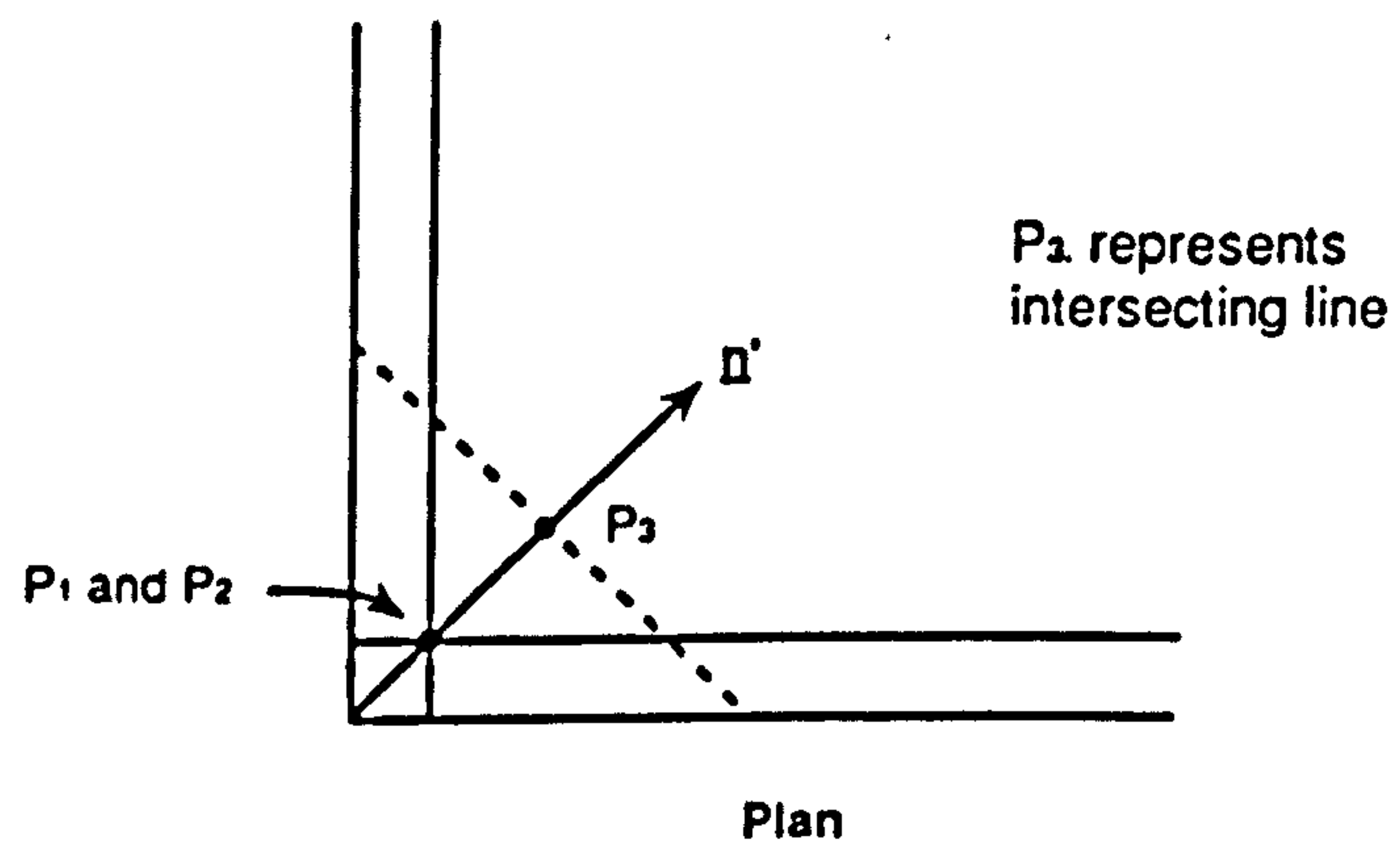
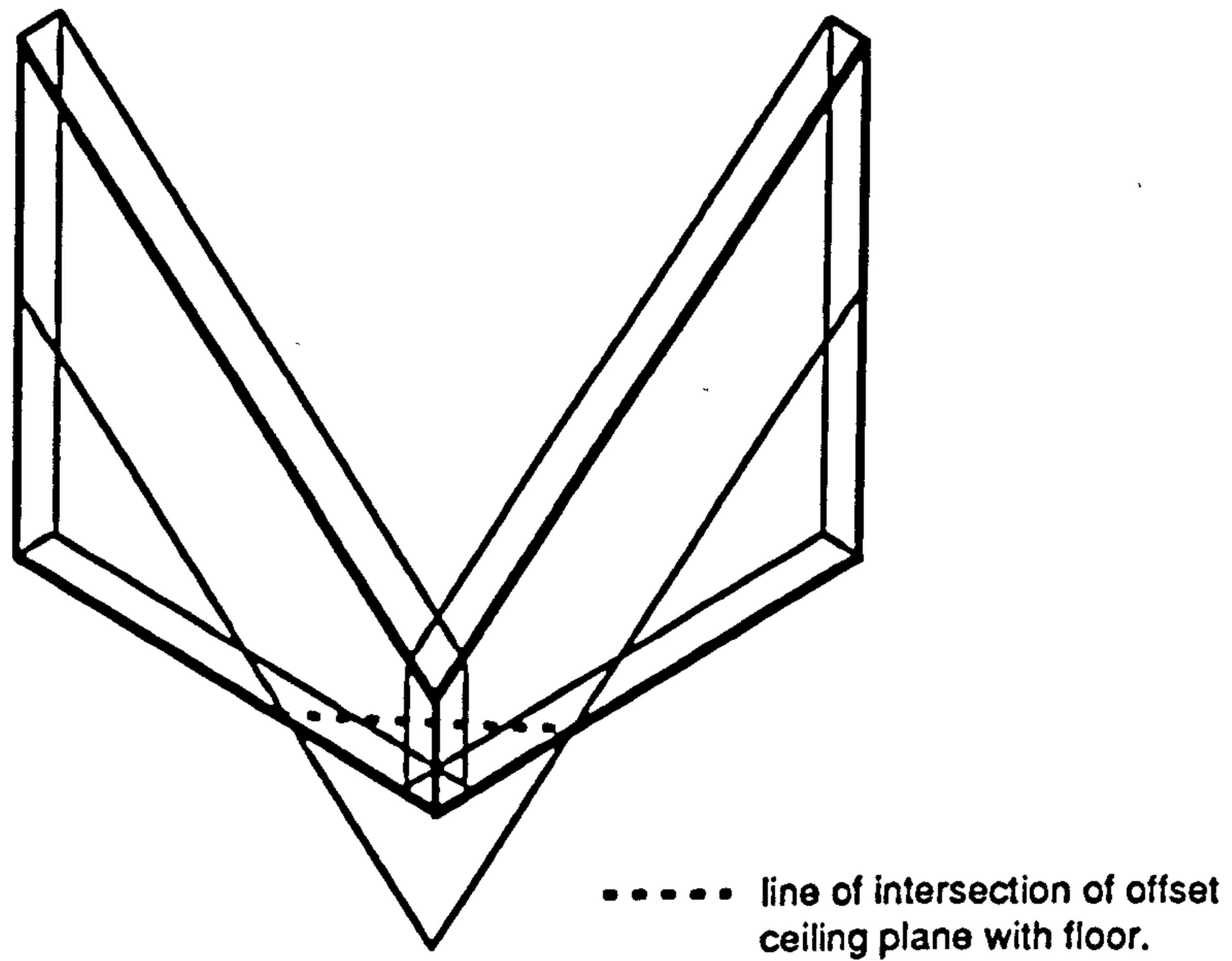


Figure 11-6 Adjustment of ceiling offset point below floor plane



A further case to consider is when there is no ceiling plane, i.e. all swept vertices intersect at a point or line. The offset point is calculated to be at the intersection of the two offset wall planes and a vertical plane through the swept vertex. Figure 11-7 illustrates a case for which this gives the correct geometrical result. However, in more complex cases, it is necessary to calculate the intersection of all the offset wall planes of the space.

c) Co-linear vertex, floor offset point:

This point is positioned at the intersection of the plane of the offset wall face, the plane of the horizontal floor face, and the vertical plane whose normal direction is that of the original graph edge. See figure 11.8. As illustrated, if the widths of the adjoining walls are not equal then two offset points are required.

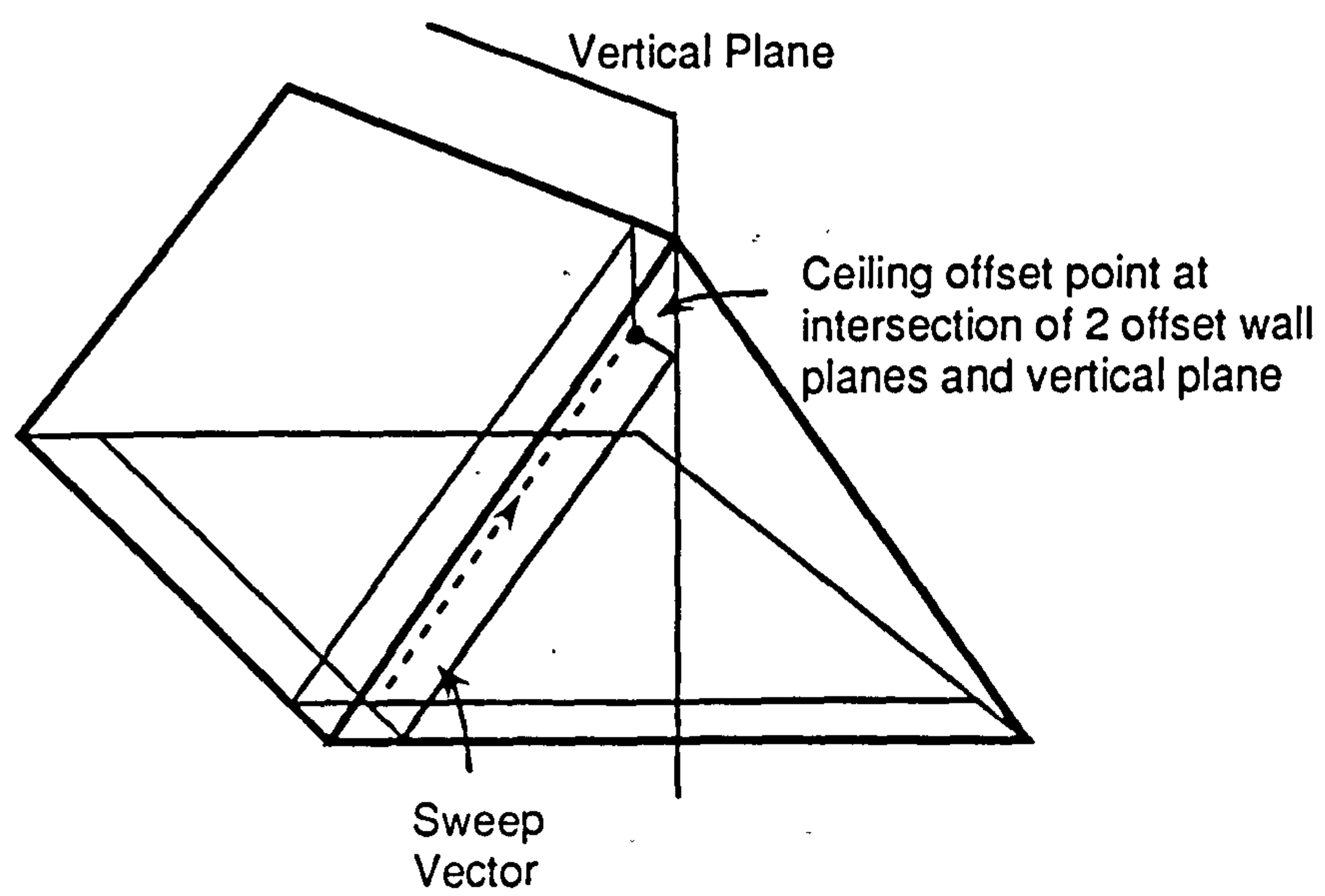
d) Co-linear vertex, ceiling offset point

This point is positioned at the intersection of the plane of the offset wall face, the plane of the offset ceiling face, and the vertical plane whose normal direction is that of the original graph edge, and is calculated as b) above. Again two offset points may be required.

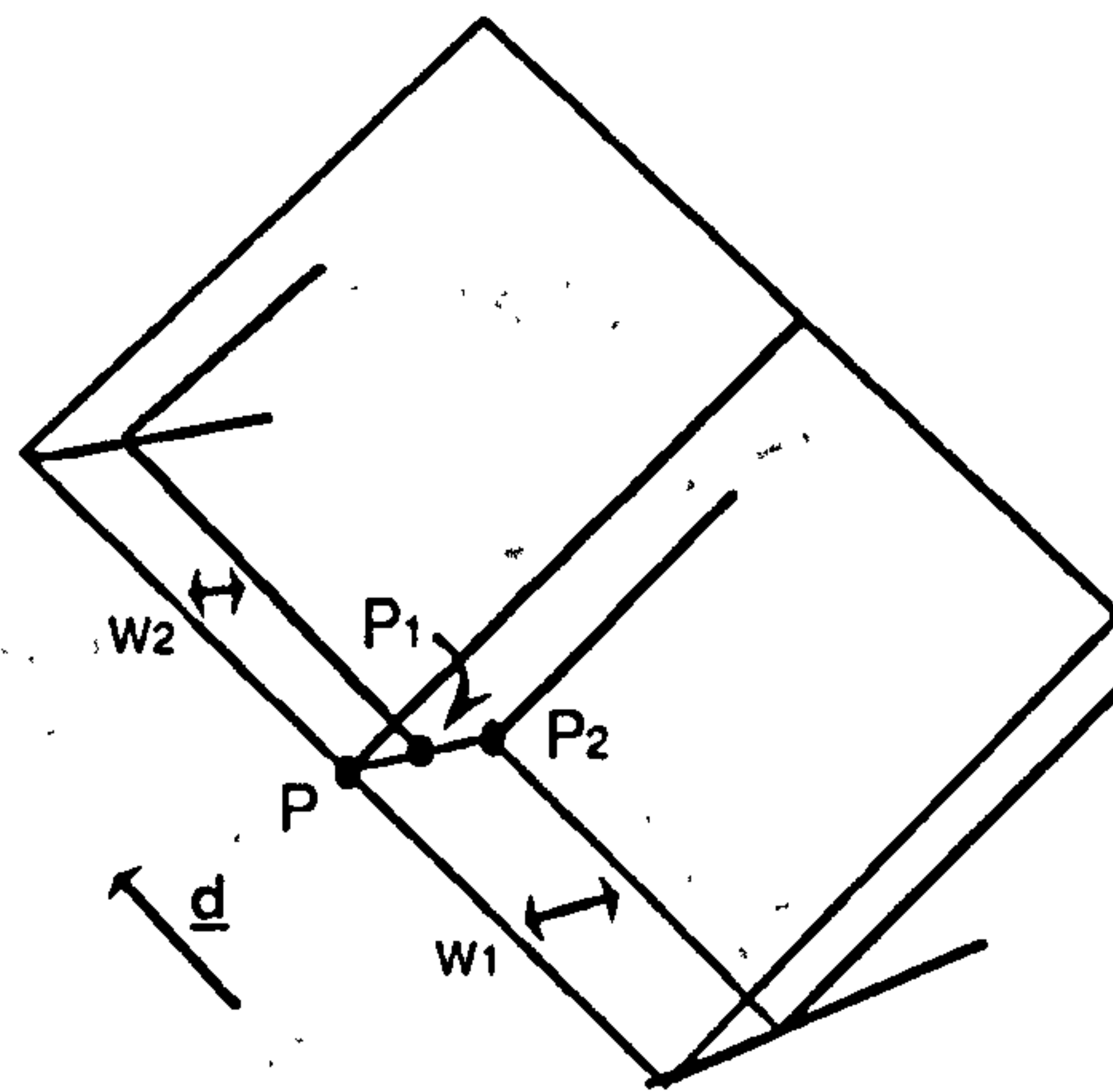
There is a special case where the swept vertex is calculated as dividing the 'up' edge of the adjoining corner vertex. The offset point is calculated by the same method as that of the point of the swept vertex: it is at the intersection of two lines, one through the floor offset point in the sweep direction and the other connects the ceiling offset points of the two corner vertices.

To summarise, the calculation of offset points is not straightforward, it requires the detection of special cases, and assumptions are made regarding their positioning. However in the majority of cases, the calculation of these internal offset points is a reasonable representation of the true building geometry. An additional field is required in the node instance entity to reference a point, the complete record definition is given in table 11-5.

An additional field is required in the wall construction entity to define the position of the offset face planes, and is shown in the complete definition in table 11-6.



**Figure 11-7** Calculation of ceiling offset point when there is no ceiling face



$\underline{d}$  direction of graph edge

P Co-linear vertex

$P_1, P_2$  Floor offset points

$w_1, w_2$  wall widths

**Figure 11-8 Floor offset point of co-linear vertex**

### NODE INSTANCE

Field Name	Description
point	3D point of node
vertex	Corresponding vertex of winged-edge structure.
offset	3D offset point of vertex
nextnode	Next node instance in list for node

Table 11-5: Node Instance Data Record referencing offset point

### CONSTRUCTION

Field Name	Description
nextcon	Next construction in list for building
lface	'Left' face of construction.
rface	'Right' face of construction.
normal	3 components of normal direction of left face.
offsettype	Type indicating positions of offset faces: 'centre', 'left' or 'right'
catlist	First of linked list of categories for construction

Table 11-6: Construction Data Record referencing offset point

As indicated above, it would be possible for a user to define an external wall with an offset external face, if the original face position corresponds to the centre line of the wall or the internal line of the wall. However, no representation is created of the external faces of constructions, i.e. there is no winged-edge representation of the 'external' space. From the current representation it is possible to determine the required construction adjacencies, from which the relevant faces can be found, and the necessary intersection point calculated. However, as there is no corresponding space vertex, there is not a node instance with which to associate the point.

To produce an accurate geometric model, therefore, the restriction is made that the 'true' position of a face, referenced by an external wall construction, should correspond to the external side of the wall.

## 11.2 Windows

There is the requirement to model windows i.e. glazed areas in walls and in horizontal and sloping roofs. They are usually part of the external envelope of a building. Windows are considered to be additional construction entities that are 'inserted' into walls and roof constructions and to which are assigned a construction type usually defining a transparent material.

The following need to be considered:

- a) the definition of a window shape,
- b) the assignment of a construction type to a window,
- c) the geometrical representation within the building,
- d) the allocation of windows to wall and roof constructions i.e. external ceiling constructions.

### 11.2.1 Window Types

As a table of Building Elements lists all the wall, floor and ceiling types within a building model, so a table of window types is defined. Each window type is then allocated a construction type from the database, and its geometrical shape is specified.

A 2D geometrical shape is required. The simplest 2D shape is a rectangle, which is the most common window shape, and can be easily defined by its two dimensions. Therefore, the present implementation allows for a set of window types to be defined, with the assumption that they are all rectangular. For the purpose of thermal analysis, it is usually adequate to approximate any window shape by a rectangular shape.

A type is then selected from this table and an instance of it is created in the plane and within the boundary of a construction. This compares to the usual CAD method, where a library of window components are defined, and instances of the components are then created to form the model.

### 11.2.2 Representation of Window Instances

A window is defined by an instance of a window type. Therefore the data of a window instance entity must include:

- a) a reference to the window type,
- b) the 3D position of the window in the model.

At the model input and editing phase, a construction must reference the instances placed within it. Such instances can then be edited i.e. deleted, and more inserted. For a 3D display of the building model a 3D representation of the window instance needs to be created i.e. a window construction. Finally for a fully evaluated building model, these window constructions would then be 'merged' into the parent construction to create the true wall and ceiling constructions.



A window construction can be represented by a rectangular lamina of two co-incident faces, i.e. a separate winged-edge data structure consisting of 2 faces is created and is referenced by the window instance entity.

A linked list of instance entities is required to represent the windows inserted in one construction. Therefore, as a construction can refer to a linked list of categories, it can also refer to a linked list of window instances.

A new data entity is therefore defined for a window instance and a reference to a linked list of these is added to the construction data entity. These are specified in tables 11-7 and 11-8.

### WINDOW

Field Name	Description
wintype	Reference to Window Type
point1	First 2D position point
point2	Second 2D position point
face	First face of window construction

Table 11-7: Window Data Record

### CONSTRUCTION

Field Name	Description
nextcon	Next construction in list for building
lface	'Left' face of construction.
rface	'Right' face of construction.
normal	3 components of normal direction of left face.
offsettype	Type indicating positions of offset faces: 'centre', 'left' or 'right'
catlist	First of linked list of categories for construction
windlist	First of linked list of window instances placed in construction

Table 11-8: Construction Data Record referencing Windows

#### 11.2.3 Creation of Window Construction

A window instance is created with walls in the 2D floor graph. As category entities are temporarily assigned to a graph edge, or graph face and then transferred to the resulting entity, a construction or space, so a window

instance is temporarily assigned. Assignment to a graph face implies windows placed in the resulting external ceiling construction of the space. The type and positioning, defined by two points are specified at this stage, the creation of the 3D window construction is included within the sweep process.

The geometry of a window construction is dependent on:

- a) the plane of the parent construction
- b) the dimensions specified by the window type
- c) the positioning points.

The derivation of the window construction differs slightly for a wall and ceiling construction and is described below.

#### 11.2.3.1 Derivation of Wall Window

The placement of a window in a wall is shown in figure 11-9. A wall window can only be positioned such that its two positioning points are 'on' a graph edge. The distance between these points is therefore the length specified for the window type. The window lies in the plane of the resulting wall construction, which may be inclined. It is positioned such that the window length is parallel to the originating graph edge, and the window height is perpendicular in the plane. The perpendicular distance between the window 'bottom' and the wall 'bottom', i.e. the position of the graph edge must be defined also. For a wall window, the floor to sill height can be considered an attribute of the window type, and is therefore included within the type definition.

The lamina of two faces representing the window construction has four vertices, each referencing a node instance, i.e. the lamina references four nodes each with one vertex. An offset point is calculated for each vertex to lie in the offset plane of the internal face of the construction. It is offset from the node point by the thickness of the construction in the normal direction of the construction.

#### 11.2.3.2 Derivation of Roof Window

The placement of a window in a roof is shown in figure 11-10. The two positioning points of a roof window can be anywhere within the graph face. The first point indicates the position of the 'bottom-left' corner of the window, if the window were projected onto the xy plane, the z co-ordinate is then calculated from the plane of the ceiling. The second point defines the orientation of the projection of the 'bottom' line of the window on the xy plane. The bottom-right corner therefore lies at a distance of  $l$ , the first window dimension, from the bottom-left corner in the ceiling plane. The second positioning point in fact often corresponds to the projection of the 'bottom-right' corner of the window onto the xy plane. The 'top' corner points of the window are therefore calculated as being at a perpendicular distance of  $h$ , the second window dimension, from the bottom line, in the ceiling plane.

Again, the window construction is represented by a lamina, and offset points are calculated dependent on the thickness of the ceiling construction type.

11.3 Zones

As discussed in chapter 3, the attribute of a space is a zone number. This requires a further extension to the space data entity as shown in table 11-9.

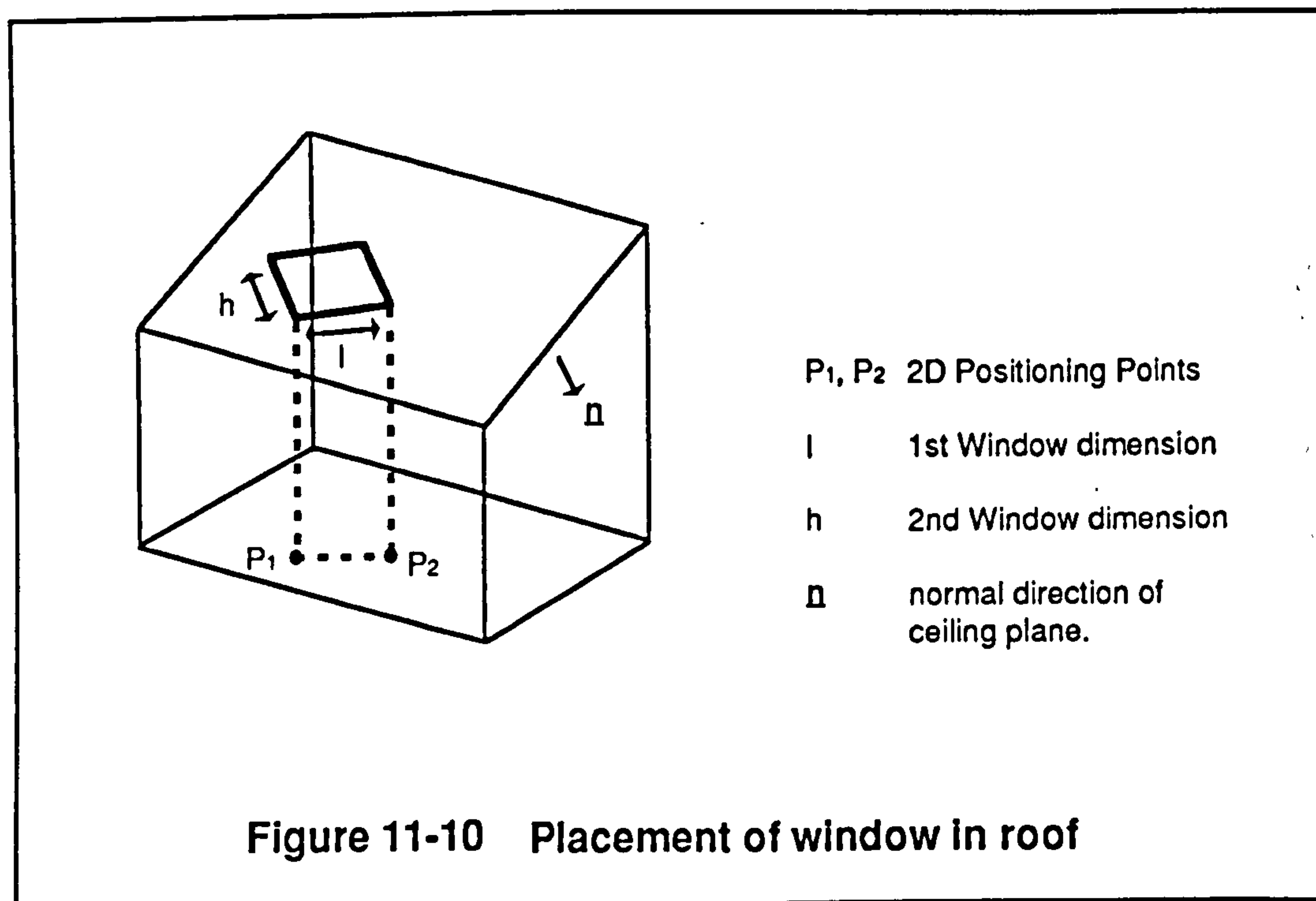
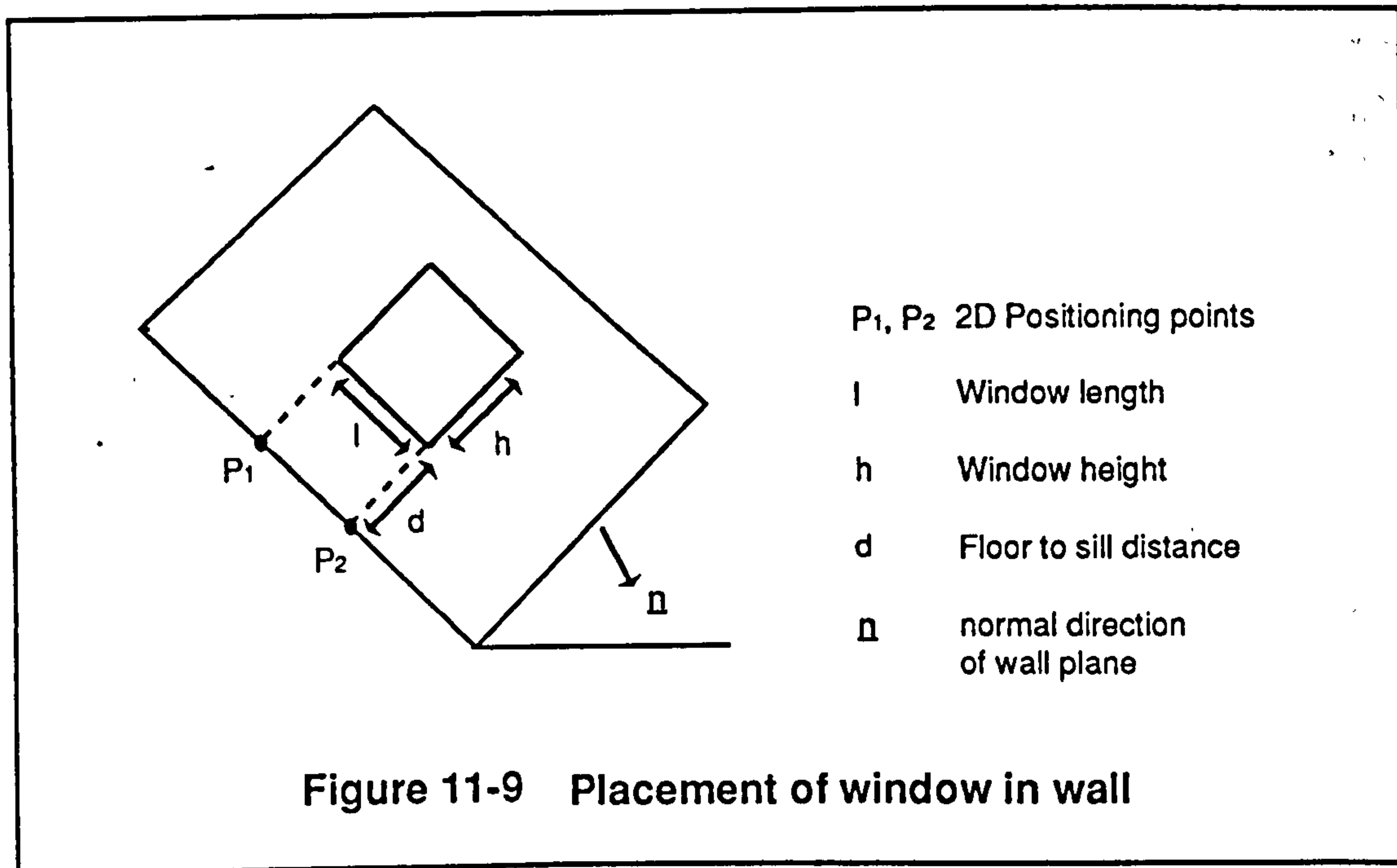
SPACE

Field Name	Description
nextspace	Next space in list for building
lowerfloor	No. of lowest floor of range of space
upperfloor	No. of highest floor of range of space
floorface	Floor face of space unit of lowest floor
ceilingface	Ceiling face of space unit of highest floor
spacelink	Space link, null if range is one floor only
floorcat	First of linked list of categories for floor face
ceilingcat	First of linked list of categories for ceiling face
zone	zone number

Table 11-9: Space Data Record referencing zone

Again, a zone number is assigned to a graph face and then transferred to the resulting space.

This chapter concludes the description of the building model representation in its part-evaluated state. A complete description of all entities of the representation is given in Appendix A.





## 12. INTERFACE TO CREATE FLOOR GRAPH

The process described in Chapter 8 to create a planar graph requires 2D line geometry to be input. A user interface must exist to allow the specification of this geometry, with options that assist the user in positioning the lines that represent the walls and the windows. The need to assign attributes to the graph edges and faces has been discussed in Chapters 10 and 11, and this functionality must also exist in the user interface.

This chapter discusses the requirements and implementation of a user interface. The form of the user interface to the software that has been developed to produce the required Building Modelling system is described in Appendix B, and examples of the graphical displays are also given. The 'graph edge' and 'graph face' data entities as defined in chapter 9 are also described. The interface to architectural CAD software is also considered here, as an alternative method of specifying the 2D geometry and its attributes.

### 12.1 Creation of 2D Geometry

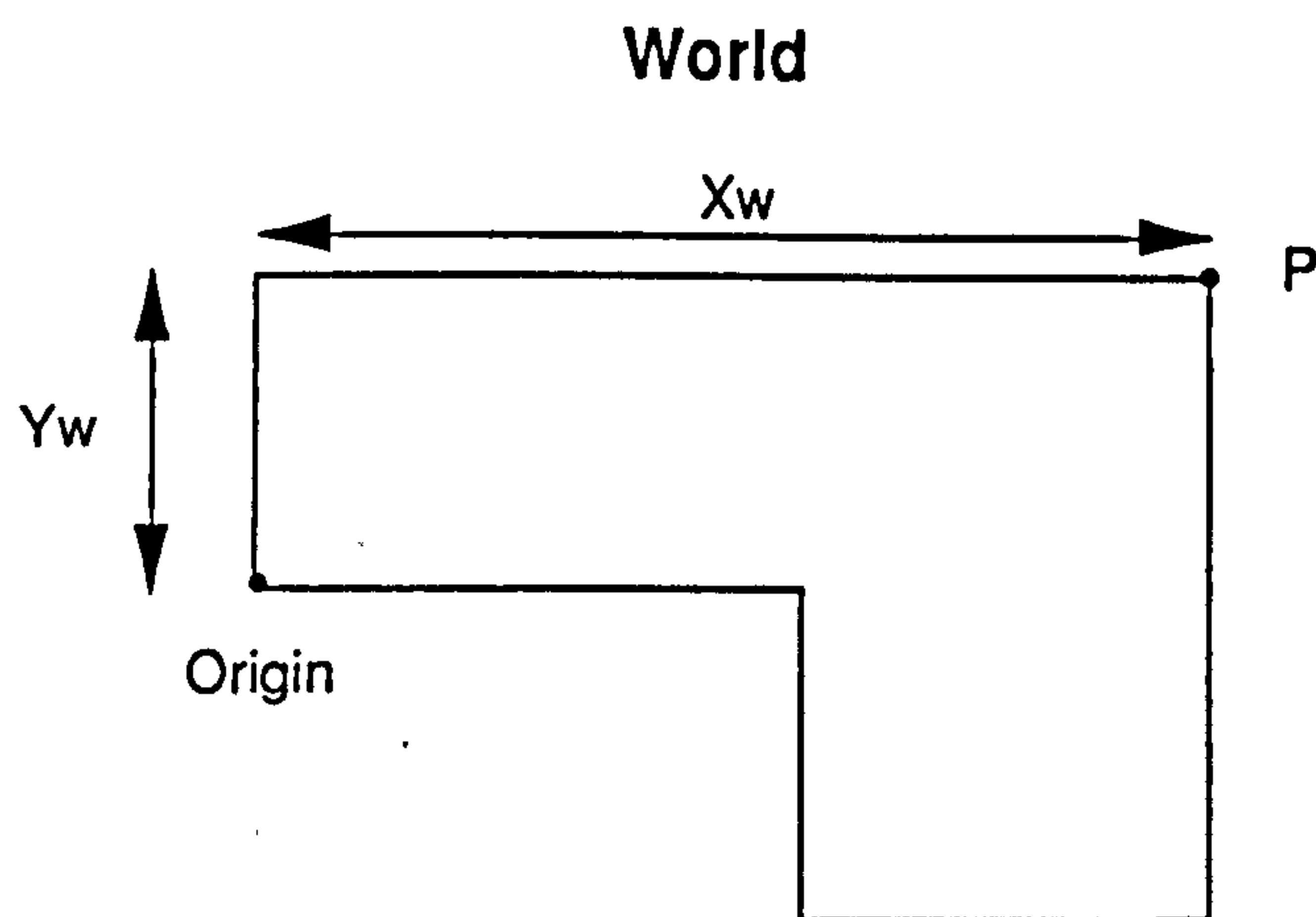
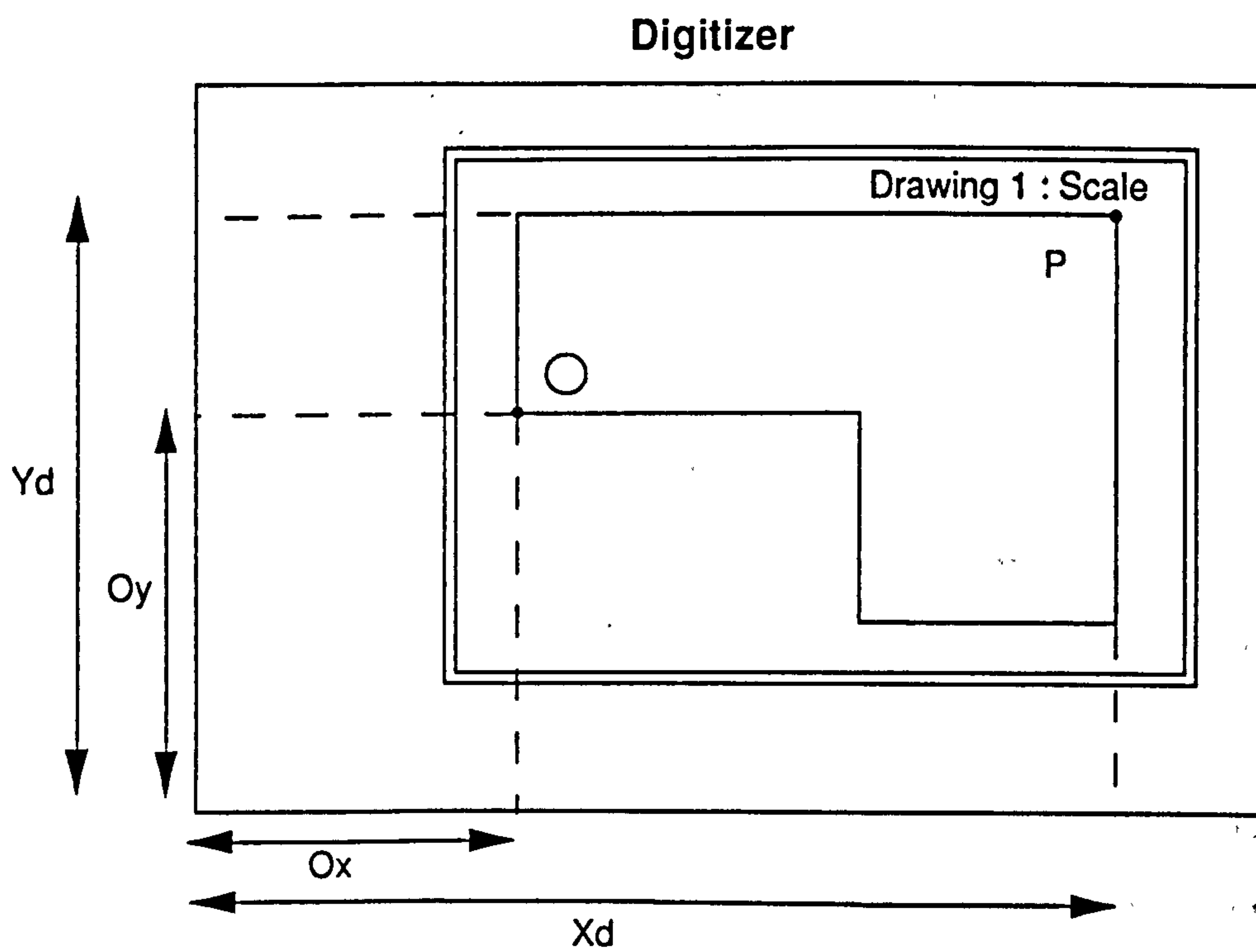
The engineer who wishes to create a building model for analysis will normally be given scaled drawings, plans and elevations that detail the layout and dimensions of the walls and windows. If this software tool is also to be used at the early stages of building design it must also be possible to sketch in the model from very little information.

#### 12.1.1 User Input of 2D points

A digitiser is ideal for tracing around scaled drawings and generating 2D points, and can also be used for sketching where there are no drawings. A digitiser generates 2D points in the digitiser co-ordinate space which is dependent on the size of the digitiser and the units used: inches or mm. A transformation mapping is necessary to convert these points into world space in which a building is modelled. This mapping is dependent on the scale of the drawing to be input, and the position of the drawing on the digitiser. See figure 12-1. Therefore to input a 2D line, a user is prompted to indicate the start point of the line on the digitiser, and then the endpoint. These points are then transformed into world co-ordinates.

It is important to provide feedback to the user on the graphics display of all user action. Therefore a 'rubberbanding' technique is useful to show the user the effect of positioning the endpoint as soon as the start point has been input. The generation of the graphics display requires that the world co-ordinates in which the model is created are transformed into screen co-ordinates. To enable the user to input quickly a set of linked lines, as when tracing around the perimeter of a building, the default position of the start point of a new line is the endpoint of the previous.





**Transformation**

$$X_w = (X_d - O_x) \text{ scale}$$

$$Y_w = (Y_d - O_y) \text{ scale}$$

**Figure 12-1 Transformation mapping from digitiser to world space**

### 12.1.2 Applying constraints to Input Points

So that precise geometry can be created, it is necessary to treat the points generated by the user via the digitiser as an approximation to the true geometry. Otherwise non-orthogonal floor plans, with walls that do not connect could be generated. User options are necessary to constrain the position of the endpoints and the direction of the line.

The usual method in CAD systems is to allow the user to define a rectangular grid to which endpoints are snapped. This ensures an orthogonal floor plan can be created, that is with walls parallel to the x and y axis, where the walls are of precise length.

When tracing in from a drawing, instead of using a grid, it is more useful to constrain the direction of the lines, allowing the lengths of the walls to be determined by 'copying' the endpoints. An option that ensures only orthogonal lines are created is therefore useful. Other options which define the direction are:

- a) the user inputs an angle of rotation, with respect to the x axis
- b) the user indicates an existing line which is parallel to the new line
- c) the user indicates an existing line which is perpendicular to the new line.

Further options to specify the position of the endpoints of the line include typing in the line length, selecting an existing line and specifying a distance along the line from an end, and/or a perpendicular distance from the line. This type of functionality and more is usually provided in standard 2D draughting software.

The process described in Chapter 8 makes adjustments to the input points, dependent on the graph tolerance when determining that a point is 'co-incident with a vertex' or is 'on an edge', and it was suggested that the value of this tolerance should be within the users control. Therefore this 'snapping' of points to an existing vertex, or onto an edge is displayed to the user as an automatic adjustment as the line is input.

However these adjustments to the 2D endpoint must also take account of the user specified constraints for the line direction, and possibly a line length. This can result in conflicting requirements, which must be resolved within the software. Therefore a set of rules are defined, and the endpoints are adjusted accordingly. These are summarised below:

- a) Any defined length is overridden if the endpoint of a line is adjusted to be 'at a vertex' or 'on an edge'.

b) If the direction of a new line is set, and the endpoint of a line is categorised as being 'on an edge', the intersection of the new line with the edge is calculated and becomes the new endpoint. See figure 12-2. If this intersection point is 'at a vertex' of the edge, then rule 3 below applies. If the new line and the edge are parallel the user is informed of an error.

c) If the direction of a new line is set, and the endpoint of the line, or the intersection point from rule 2 above is categorised as being 'at a vertex', the action taken then depends on the position of the starting point of the line:

i) The starting point is not at a vertex, or on an edge: the new line is offset to a parallel position. See figure 12-3.

ii) The starting point is on an edge: the starting point is adjusted along the edge. See figure 12-4. If this point is 'at a vertex' of the edge iii) below applies.

iii) The starting point is 'at a vertex' of the edge, that is both endpoints are 'at a vertex' and the direction of the line has been specified. This can often occur when a loop is completed, as when tracing a perimeter, with all input lines drawn orthogonally. See figure 12-5. It is checked whether the starting vertex has only one existing edge connected to it, which is not parallel to the new line. If so, this edge is extended and the starting vertex is adjusted to be at the intersection of this edge and the new line. See figure 12-6. If this is not possible a similar attempt is made to adjust the end vertex. The user is informed of an error if this also fails.

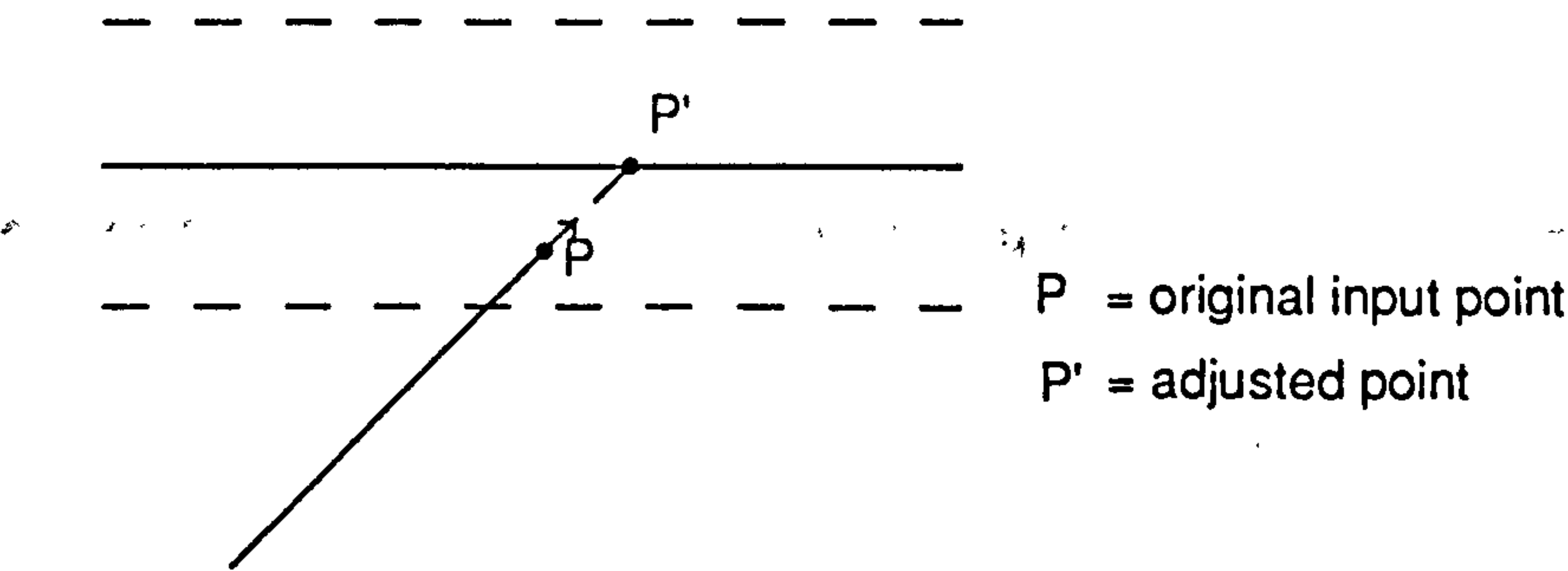
To summarise, the direction of a line, if specified, is always adhered to, generating an error if it is not possible to create the required line.

The above defines the two endpoints of a line that are input into the process to create the edges of the planar graph. This process also detects an error if the line passes 'near' to a vertex, that is an intersection with an edge is found, which is categorised as being at the vertex of the edge.

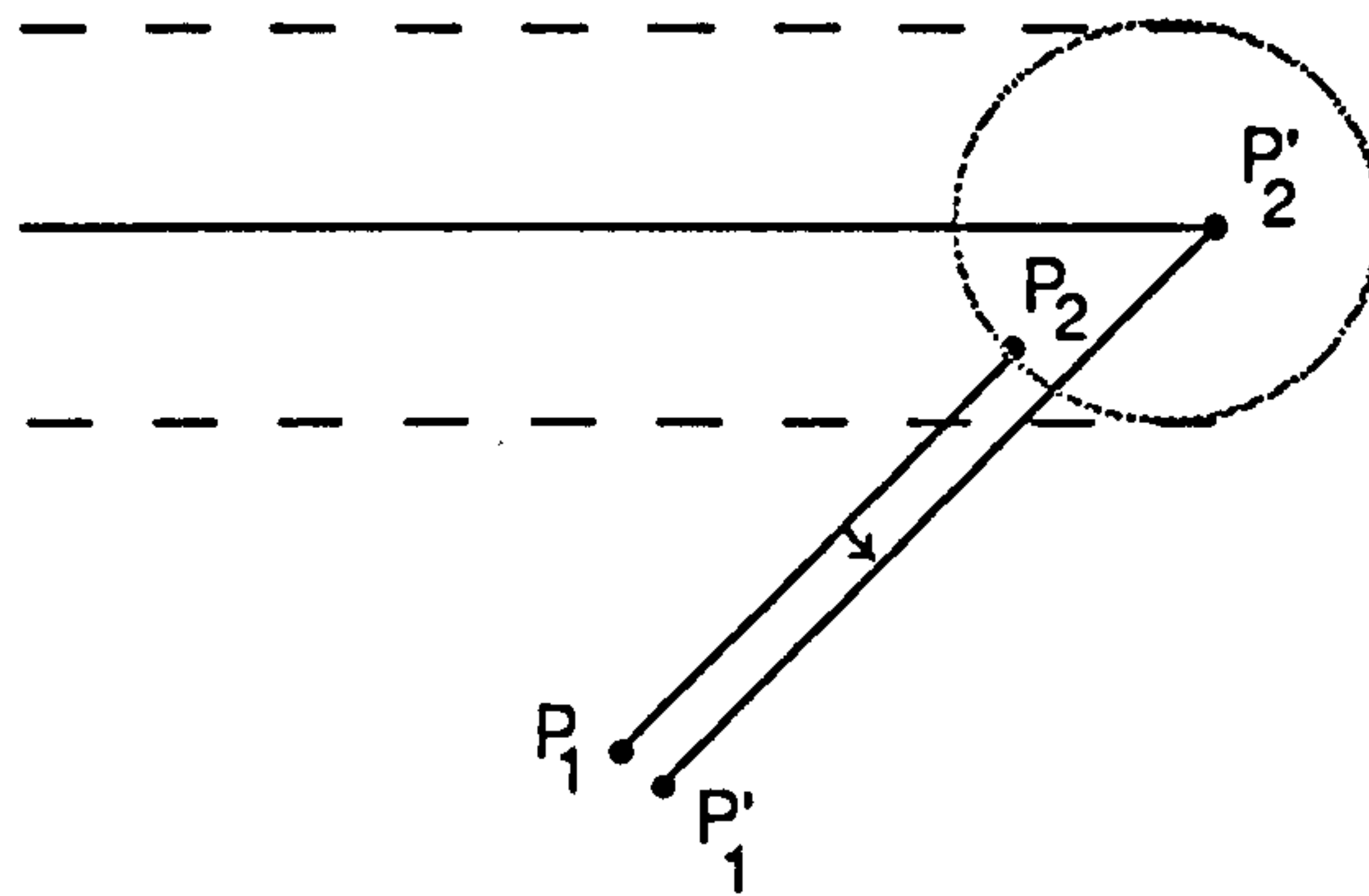
### 12.1.3 Window Positioning

As described in Chapter 11, both wall and roof windows are positioned by two input points. The user options described above are also useful to specify these points. The user selects a window type from a predefined table of types, the length is determined by this window type. Although it is not necessary to 'snap' to the vertices and edges in the manner described for 2D lines, there are some restrictions on where the windows are placed.

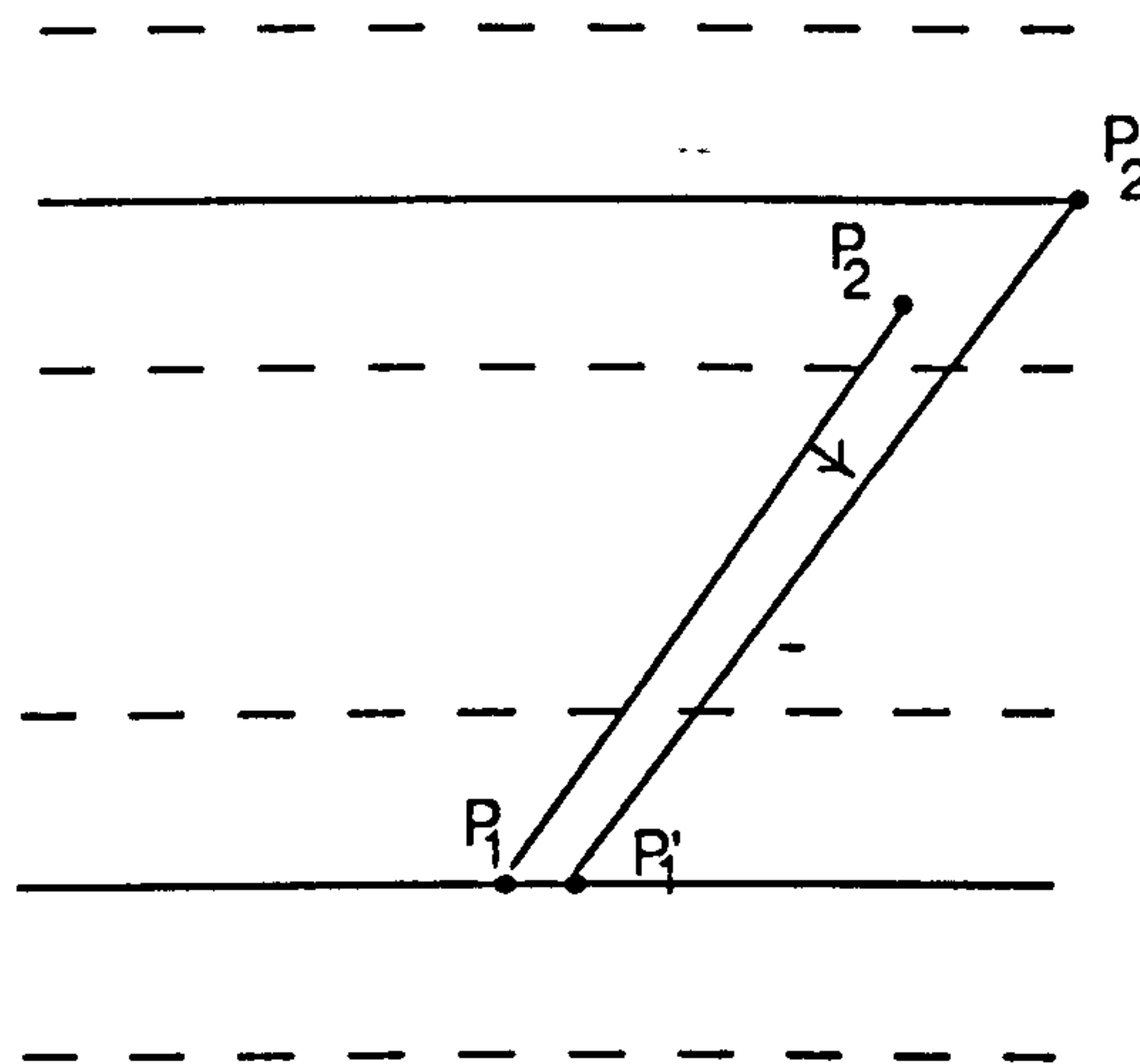
Obviously a wall window can only be placed in a wall. This means that the endpoints are adjusted to be 'on an edge', and that both must lie on the same edge.



**Figure 12-2      Adjustment of endpoint 'on an edge' when direction is set**

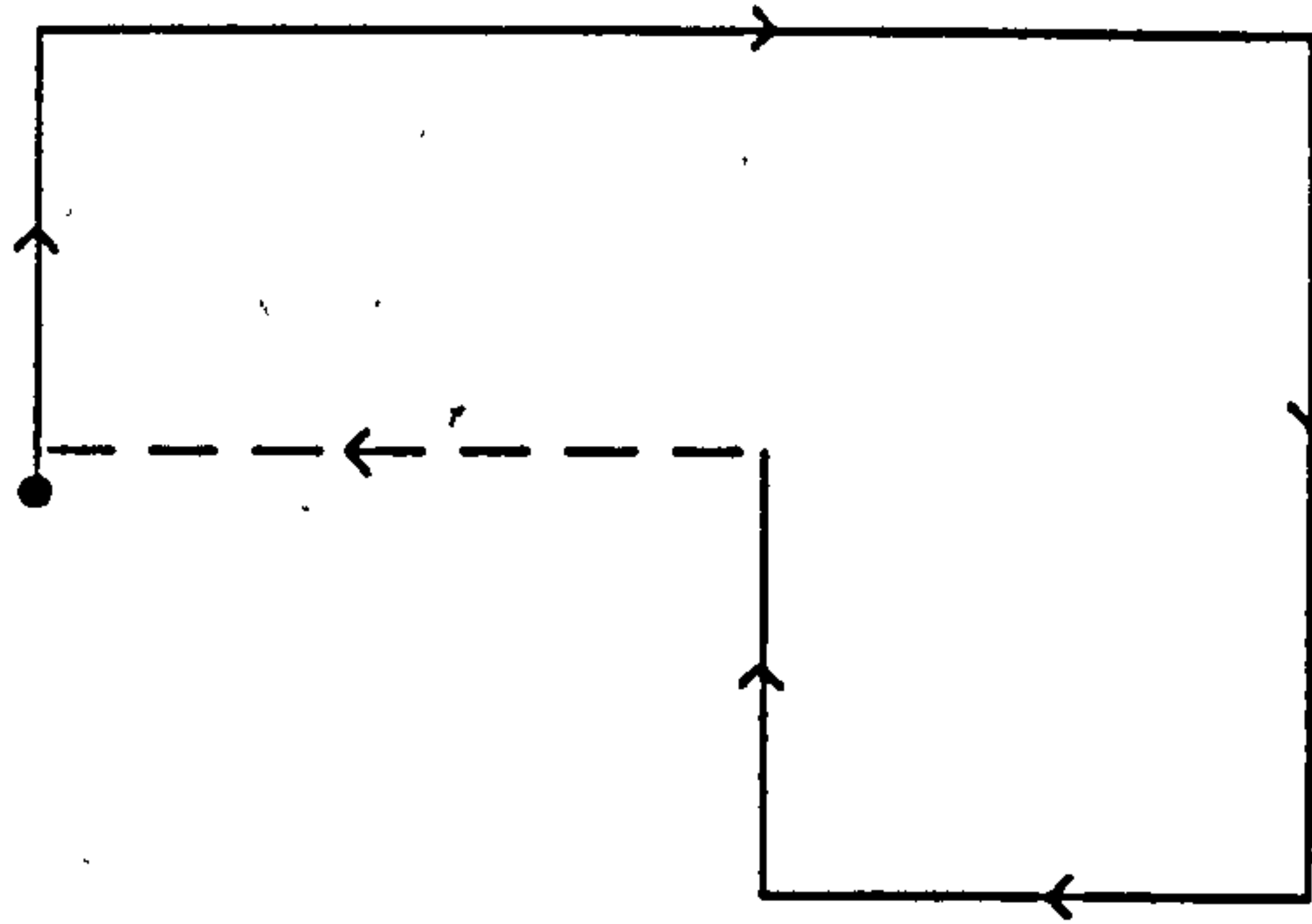


**Figure 12-3** Adjustment of line to parallel position when endpoint is 'at a vertex' and direction is set

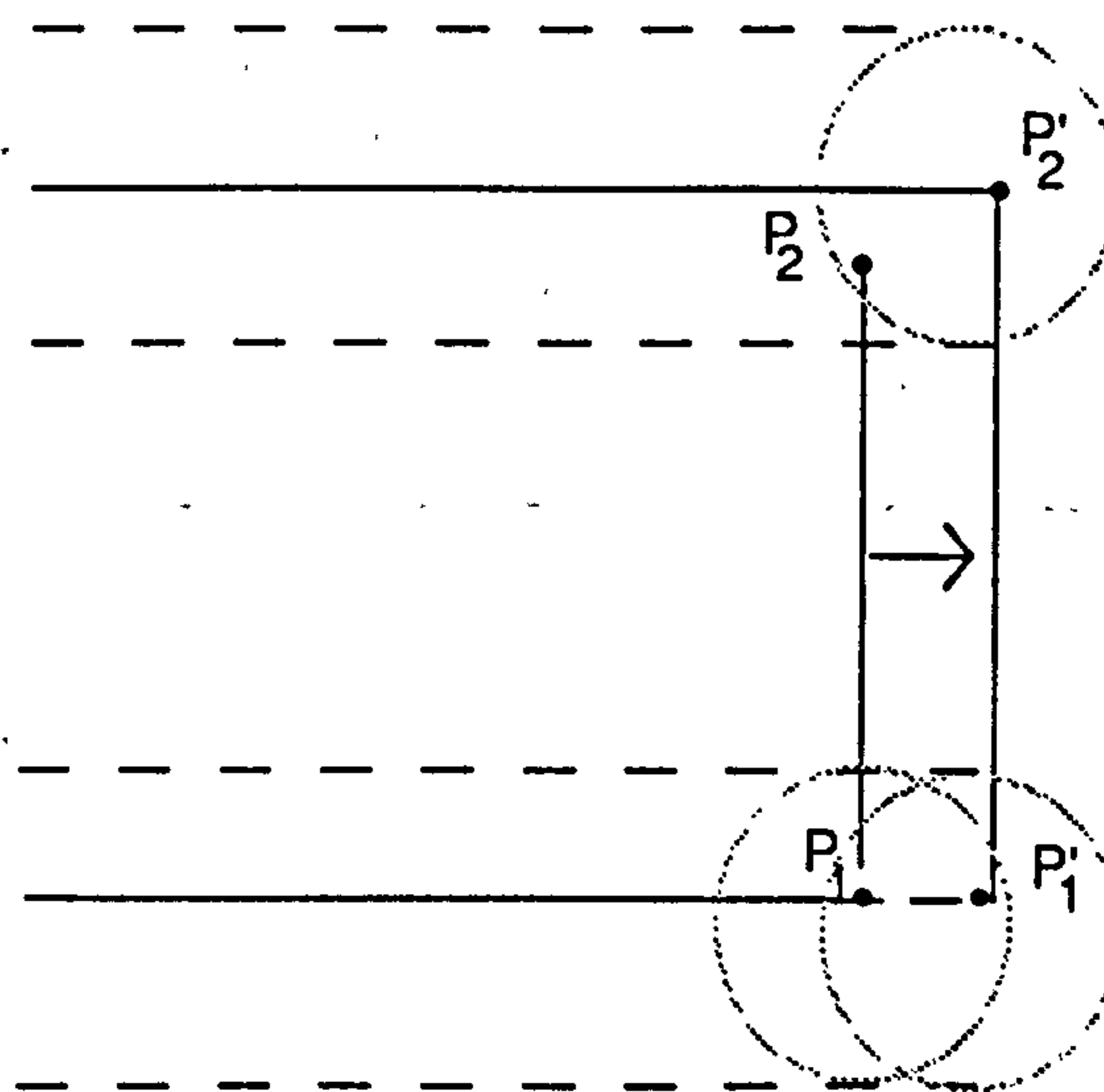


**Figure 12-4** Adjustment of line to parallel position when endpoint is 'at a vertex', startpoint is 'on an edge' and direction is set





**Figure 12-5** Mismatch of endpoints when completing a loop with orthogonal lines



**Figure 12-6** Extension of existing edge and adjustment of line to parallel position when both end points are 'at vertex' and direction is set

A roof window is placed in a space ceiling, that is both endpoints must lie within, or be on the boundary of the same graph face. The graph face cannot be 'external', i.e. a courtyard.

## 12.2 Graph Edge and Graph Face Data Entities

Graph edge and graph face data entities were introduced as being necessary in Chapter 9, to allow the winged-edge data structure to be used for the representations of both a space and a 2D planar graph. Chapters 10 and 11 have discussed the geometrical attributes of the graph edges and faces that are used in defining the geometry of the derived constructions and spaces, and the model attributes that are transferred to the constructions and spaces for analysis to be possible.

In addition, there are some data that are preset when the graph entities are generated from the building model when walls and spaces already exist for the specified floor. These determine the operations that are valid for the entities. The 'operations flag' in the graph edge data entity indicates which geometrical operations are valid:

- a) All operations are valid: deletion, and setting geometry attributes
- b) Setting geometry attributes, but not deletion, as for an edge of a space unit linked to the floor below.
- c) No geometrical operations or deletion, as for an edge of a space unit linked to both the floor above and below.

The definitions of a 'graph face' data entity, referenced by a face entity, and a 'graph edge' data entity, referenced by a edge entity are given below, in tables 12-1 and 12-2 where the category for each field indicates whether it is a geometry attribute, a model attribute, or an attribute that is preset.

### GRAPH FACE

Field Name	Description	Data Category
external	Flag 'on' indicates the face represents an external space	Geometry
floorcat	First of linked list of categories for space floor face	Model
ceilingcat	First of linked list of categories for space ceiling face	Model
zone	zone number	Model
lowspacelink	link to lower spacelink of space or null	Set
upspacelink	link to upper spacelink of space or null	Set

Table 12-1: Graph Face Data Record

### GRAPH EDGE

Field Name	Description	Data Category
height1	Height assigned to 'next' vertex w.r.t face of right loop	Geometry
height2	Height assigned to 'previous' vertex w.r.t. face of left loop	Geometry
inclination	Angle of inclination of wall	Geometry
offsettype	Type indicating positions of offset faces: 'centre', 'left' or 'right'	Geometry
catlist	First of linked list of categories for construction	Model
windlist	First of linked list of window instances placed in construction	Model
opflag	Flag indicating valid operations on edge: all, geometry, none	Set

Table 12-2: Graph Edge Data Record

### 12.3 Assignment of Attributes

User options must exist to allow the assignment and modification of both geometrical and model attributes to the graph edges and faces. A method of 'picking' is required for the user to identify an edge or a face for these assignments. This is achieved by the user indicating a point on the drawing on the digitiser. The point is then interpreted to find the nearest edge, if an edge must be selected, or the face within which the point lies, if a face is being selected. To confirm selection, the entity must be highlighted on the display. This is achieved by over-drawing an edge in red, or by filling in the polygon in red.

Note that it is not the original input lines that are selected, but the resultant graph edges. This gives the user greater flexibility in creating the model: an individual edge can be selected and deleted, and different assignments can be made to each edge section of the original line. However this has the disadvantage that multiple assignments may be necessary to define the attributes of one input line corresponding to a wall. However if assignments are made when a wall is input, new walls created from splitting the wall will have the same assignments so this can be minimised. Therefore multiple assignment is only necessary when a line intersects with several spaces and

is divided into many edges as soon as it is input. The management of these attributes within the software as the Euler operations create and delete edges and faces must be performed in a consistent manner and is discussed later.

The implementation of the user interface to allow the assignment of each attribute is now discussed.

### 12.3.1 Height

It is necessary to be able to assign a height attribute to each vertex with respect to each of its adjoining faces. The default height, that is if no height is assigned, is the floor to floor height: the difference between the position of the current floor and the position of the next upper floor.

Instead of selecting each vertex in turn, multiple assignments can be achieved in a number of ways:

- a) By space: all vertices on the boundary of a graph face with respect to a set of selected faces. See figure 12-7.
- b) By edge: either its two vertices with respect to both adjacent faces, or to only one selected face. See figure 12-8.
- c) By vertex: either with respect to all adjacent faces, or to a set of selected faces. See figure 12-9.

An option is necessary, therefore to be used in conjunction with the above assignment options, that allows a set of faces to be defined.

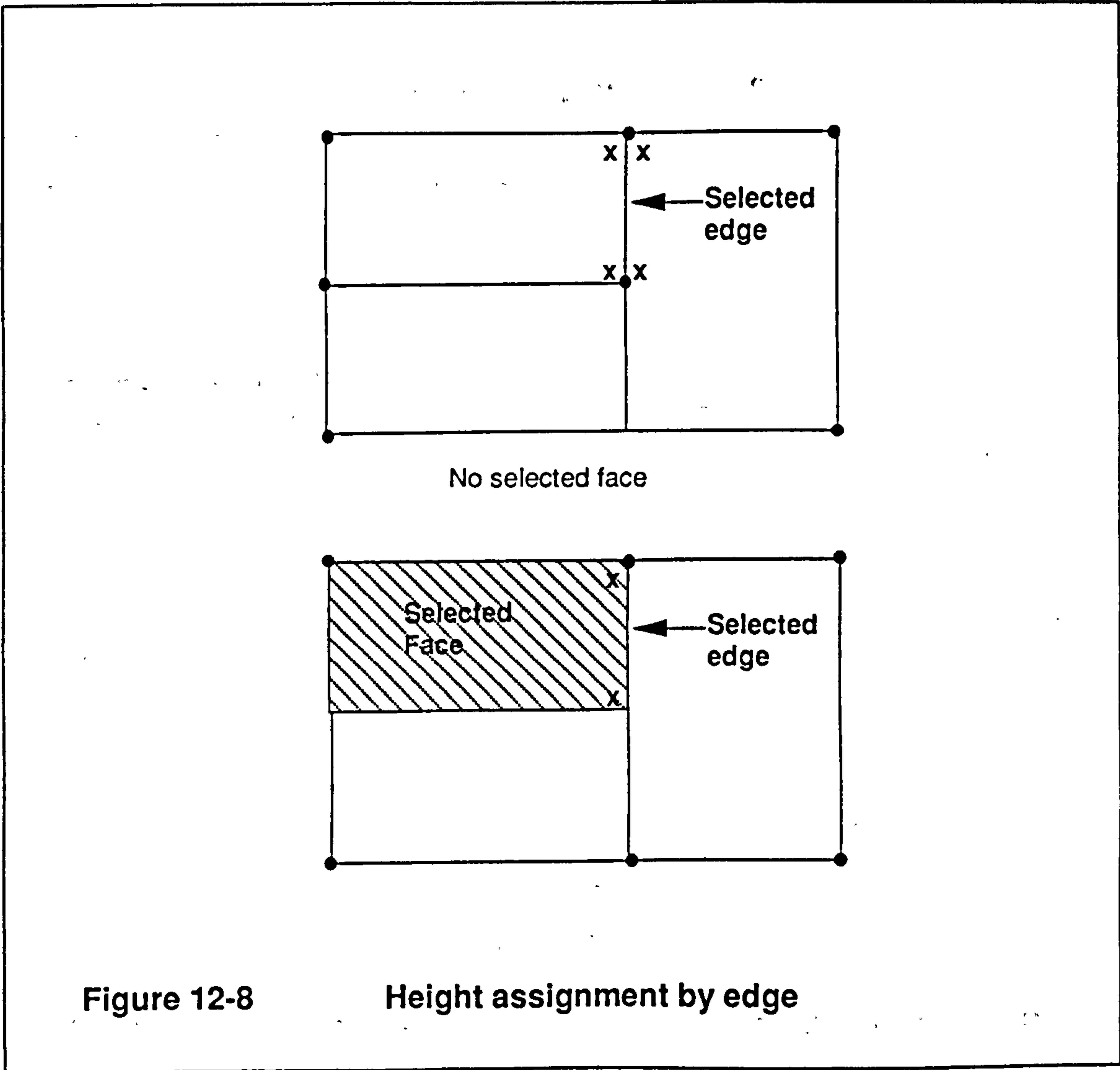
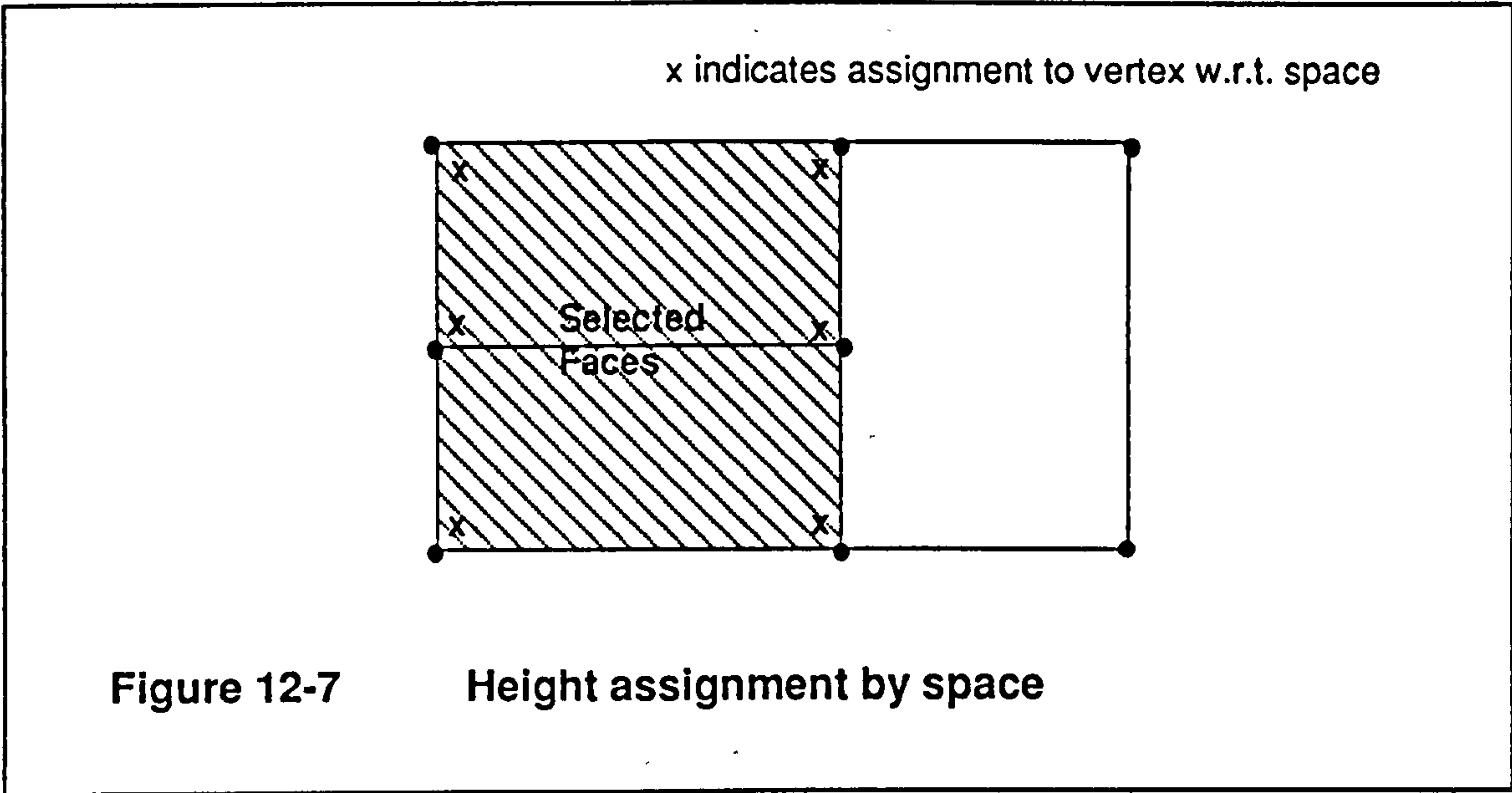
The assigned height can also be defined in more than one way:

- a) A height value is typed in
- b) A plane is defined in 3D space, from which the height of any vertex is determined by projecting a line perpendicular to the xy plane up to the plane. The plane can be defined in two ways: either by defining 3 points, by selecting 3 vertices each with an assigned height, or by selecting an edge, representing a horizontal line, and typing in an inclination. See figure 12-10.

### 12.3.2 Inclination

An angle of inclination can only be assigned to an 'external' edge, that is one that is adjacent to an external face, either the outer graph face, or an inner graph face flagged as 'external'.







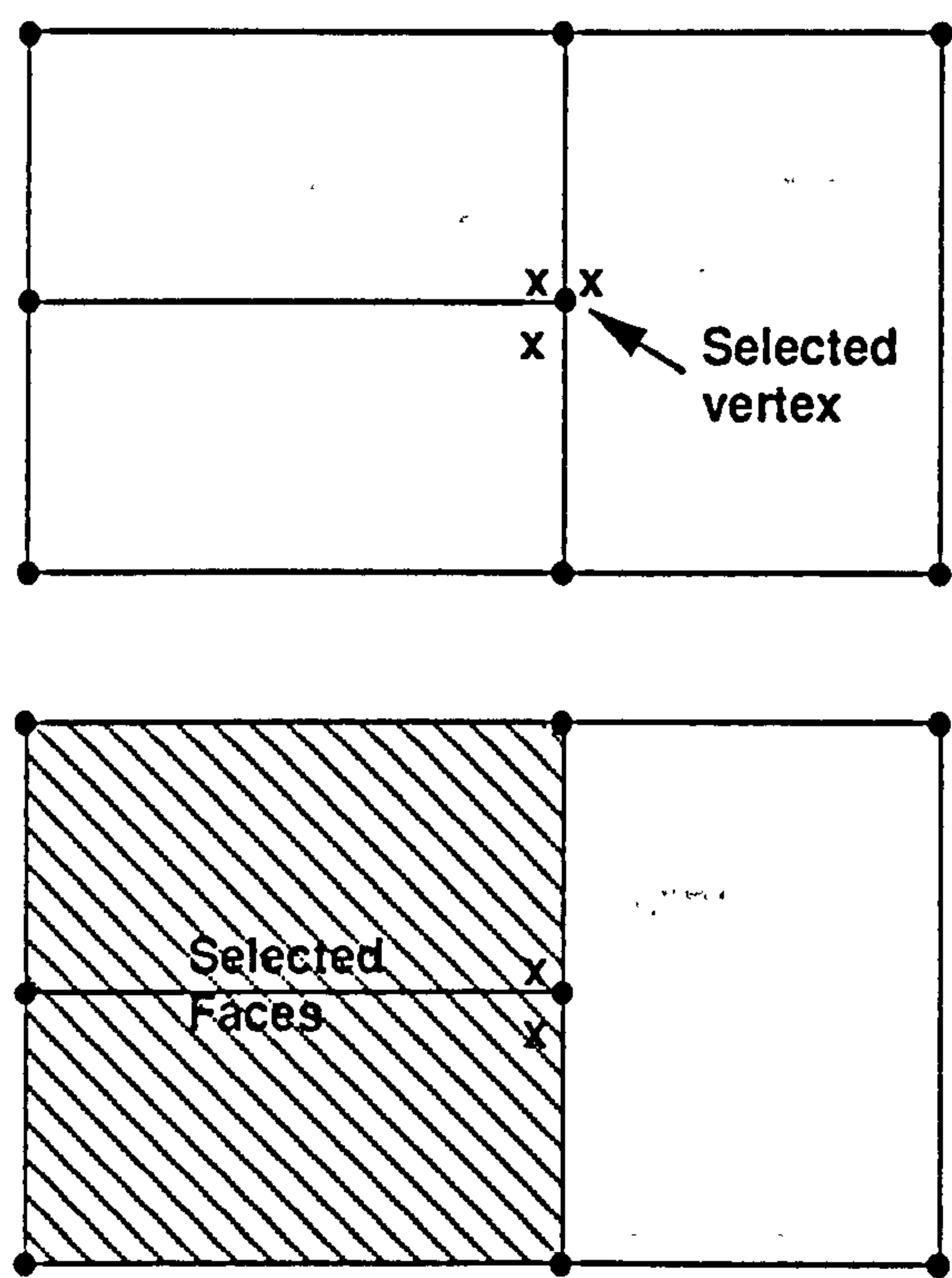


Figure 12-9      Height assignment by vertex

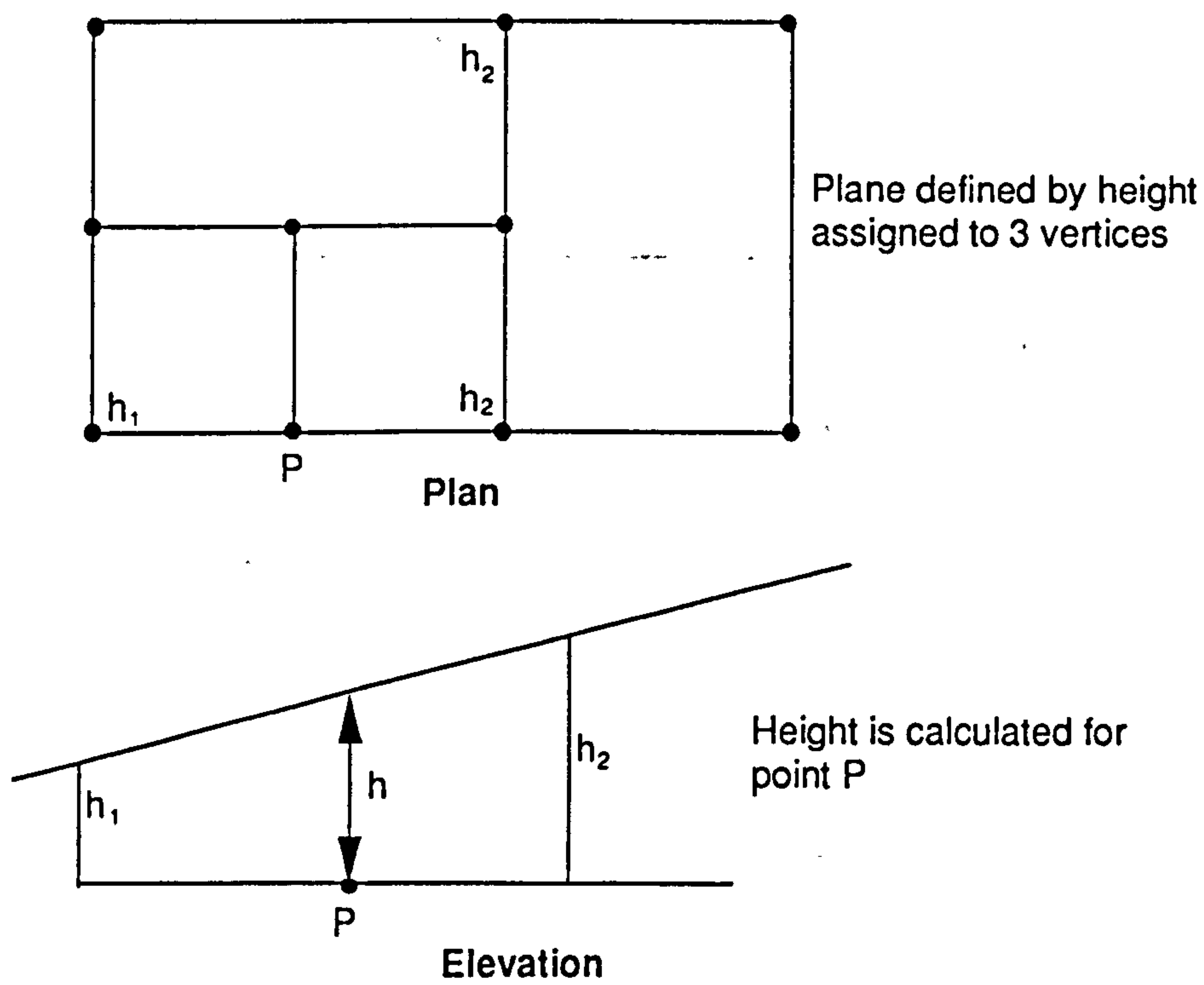


Figure 12-10      Calculation of height of vertex from defined plane

### 12.3.3 Offset Type

The offset type of an edge is defined by a current setting when an edge is created. This attribute determines how the edge is displayed: a solid line is drawn to represent the position of the input line and the resulting graph edge, and one or two parallel dashed lines are drawn to indicate the side or sides that the offset planes will be positioned. The width of this wall representation is the graph tolerance value, so that it is apparent how near a point must be input for it to be automatically adjusted onto the edge.

An option is also provided to change the offset type of an existing wall to the current setting.

An additional 'offset type' is defined, that specifies an input line represents an 'invisible wall construction'. This is displayed as a single line.

### 12.3.4 External Flag

Selection of a face acts as a switch within this option:

Space to external, or

External to space

All current 'external' faces are highlighted initially, with selection turning the highlight 'on' or 'off' accordingly. A face cannot be set external if it represents a space linked to an upper or lower floor. The spacelink entities in the Graph Face are used to check this. Neither can it be set if it is bounded by edges with an offset type of 'invisible wall construction'.

### 12.3.5 Zones

The assignment of zones may be changed for different analyses after the building geometry is complete. It is important to show any current assignment. This is achieved by associating a different colour with each zone number, and displaying the faces filled in with the appropriate colour for any assigned zone. Any faces not coloured are not assigned to a zone, indicating that the associated space would not be included if the model was analysed.

A key of colours against a zone number is displayed, and one is selected. Selection of a face then acts as a switch:

not currently assigned,

or assigned with a different zone -> current selected zone is assigned

current zone is already assigned -> not currently assigned

If a face is flagged as external, a zone number cannot be assigned.

Any adjacent faces connected by an 'invisible' edge must by definition represent spaces of the same zone. Also if a space extends upwards or

downwards, adjacent spaces connected on these floors by 'invisible' walls must also be of the same zone. The assignment to these faces or spaces is therefore automatically initiated.

### 12.3.6 Construction Types

A table of building elements is created by the user, although it is initialised with the default names described in Chapter 11. A building element name is selected from this table, in conjunction with an option indicating either walls, floors or ceilings are to be assigned. All such entities currently assigned or defaulting to the selected building element are then highlighted. A further option switch indicates whether assignment is to define the main category of the entity or a sub-category. In this latter case a proportion is also defined with each assignment, entered as a percentage.

Selection of an entity then causes a new category record to be created referencing the selected building element. This record is then either added to an existing linked list for the entity or becomes the first in the list.

### 12.4 Manipulation of Attributes with Euler Operations

Whenever a new graph edge or face is created by an Euler operation, all of the attributes described above must be initialised within the Graph Face and Edge entities. This must take into account whether a new edge is being created by an add edge operation, when attributes of adjoining edges may be considered, or by a 'Split edge' operation, when the attributes of the original edge are considered. Similarly the attributes of the original face must be considered when a face is created from dividing an existing inner graph face.

The operations that delete entities, by merging edges or by the deletion of an edge that causes two faces to be 'merged', must consider the current assignment of attributes, check whether the operation is therefore valid, and if so derive the resultant attributes. A 'Delete Wall' option is available to the user, that results in the following Euler operations, dependent on the vertices of the edge, and the adjacent faces:

- a) Kill an edge and a vertex: if the 'next' vertex of the edge has no edges attached. If the 'previous' vertex also has no edges attached, a 'Kill vertex, hole loop' operation is also necessary.
- b) Kill an edge and a face: if the left and right adjacent faces differ.
- c) Kill an edge, add a loop: if the left and right adjacent faces are the same, i.e. the edge is a wire edge.

A 'Merge Edge' operation is possible when an edge is deleted that leaves a vertex with only two colinear edges attached. If the attributes of these edges are consistent the edges are automatically merged. Split and merge wall options are also provided to the user to allow 'Split Edge' and 'Merge Edges' operations to be within the users control.

The following tables, 12-3 to 12-11 summarise the resulting attributes after an Euler Operation.

Euler Operation	Result
Add edge and vertex	height1: set to default height2: as set in cw edge for vertex wrt face of new edge (Figure 12-11)
Add edge and face or kill loop	height1: as set in cw edge for vertex wrt face height2: as set in cw edge for vertex wrt face (Figure 12-12)
Split edge	height1 of original edge: interpolated from heights of vertices wrt right face height2 of original edge: unchanged height1 of new edge: height1 of original edge height2 of new edge: interpolated from heights of vertices wrt left face (Figure 12-13)
Merge edge	Merge operation allowed only if heights at vertex wrt each face, left and right, are in line with opposite vertices.

Table 12-3: Height Attribute Assignment to Graph Edge

Euler Operation	Result
Add edge and vertex	Set to default - vertical
Add edge and face or kill loop	Set to default - vertical
Split edge	As original edge
Merge edge	Merge operation allowed only if inclinations of both edges are equal

Table 12-4: Inclination Attribute Assignment to Graph Edge

Euler Operation	Result
Add edge and vertex	Set to current user defined type
Add edge and face or kill loop	Set to current user defined type
Split edge	As original edge
Merge edge	Merge operation allowed only if types of both edges are equal

Table 12-5: Offset Type Assignment to Graph Edge



Euler Operation	Result
Add edge and vertex	Set to default - Null pointer to linked list
Add edge and face or kill loop	Set to default - Null pointer to linked list
Split edge	Linked list of original edge is copied to create a new list, referenced by the new edge
Merge edge	Merge operation allowed only if main categories of both edges are equal, linked lists are merged

Table 12-6: Wall Category Assignment to Graph Edge

Euler Operation	Result
Add edge and vertex	Set to default - Null pointer to linked list
Add edge and face or kill loop	Set to default - Null pointer to linked list
Split edge	Edge cannot be split within a window length. A new linked list is created and the original modified by moving the records of windows placed within the length of the new wall to the new list.
Merge edge	Two lists are merged together.

Table 12-7: Wall Window Assignment to Graph Edge

Euler Operation	Result
Add edge and face	First face of graph: initialised to zero Subsequent faces: copied from face being divided
Delete edge and face	Operation only allowed if zone numbers of left and right face are equal

Table 12-8: Zone Number Assignment to Graph Face

Euler Operation	Result
Add edge and face	Always set false: an internal space
Delete edge and face	Dependent on current settings of left/right faces: internal/internal: result is internal - false external/external: result is external - true internal/outer graph face: external by default internal/external: internal - false

Table 12-9: External Flag Assignment to Graph Face



Euler Operation	Result
Add edge and face	Linked list of original face is copied to create a new list, referenced by the new face
Delete edge and face	Linked lists are merged, main category of right face is ignored.

Table 12-10: Floor/Ceiling Categories Assignment to Graph Face

Euler Operation	Result
Add edge and face	Face cannot be divided through a window rectangle. A new linked list is created and the original modified by moving the records of windows placed within the new face to the new list.
Delete edge and face	Linked lists are merged.

Table 12-11: Floor/Ceiling Windows Assignment to Graph Face

## 12.5 Floor Editing Process

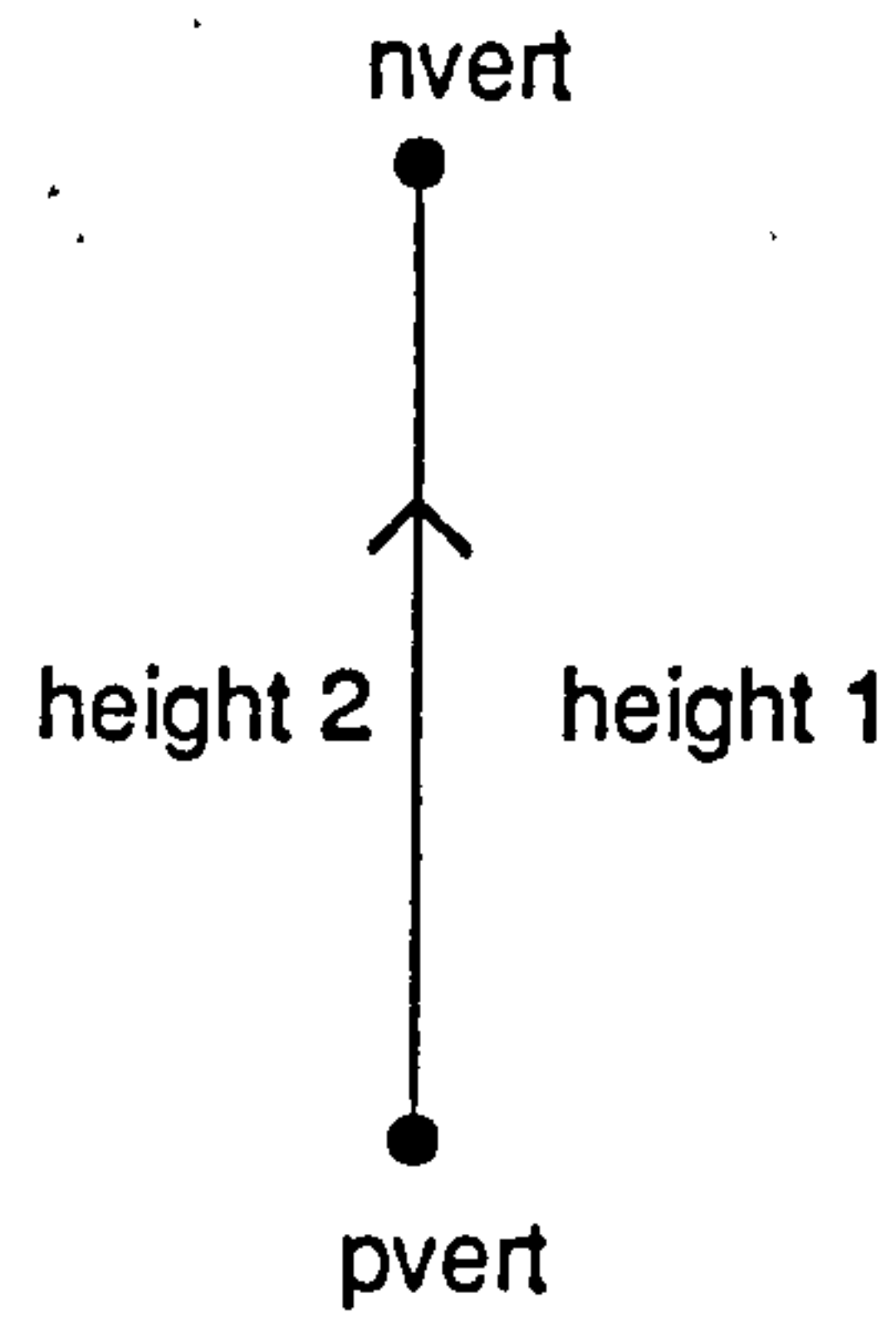
So far, the features of the user interface to create one floor have been considered. The functionality to combine individual floors into one building model must also exist. Each floor in a building needs to be identified by a number, and optionally given a description. A z position must also be assigned to each floor, the differences between these positions of adjacent floors are then the default heights of the floors. It must be ensured that the required z heights are available before a floor can be input, and conversely that the building model does not already extend to a specified z height. In this case the extended spaces will not have been divided into the vertical units at the required z positions.

### 12.5.1 Defining a Digitiser Mapping

When a floor is selected to be input or modified, a mapping from digitiser to world space must be defined.

For the first floor of a building to be input, there is no 'world' to map to. Therefore any arbitrary point within the drawing on the digitiser can be chosen to be mapped to the world origin. The user does not need to be aware of where the world origin is. The scale of the drawing is defined either by a number e.g. 100 for a scale of 1 to 100, or by digitising two points, and typing in the distance represented between the two points.

The following figures show the assignment of two heights to an edge



height 2 is height of pvert wrt left face  
height 1 is height of nvert wrt right face

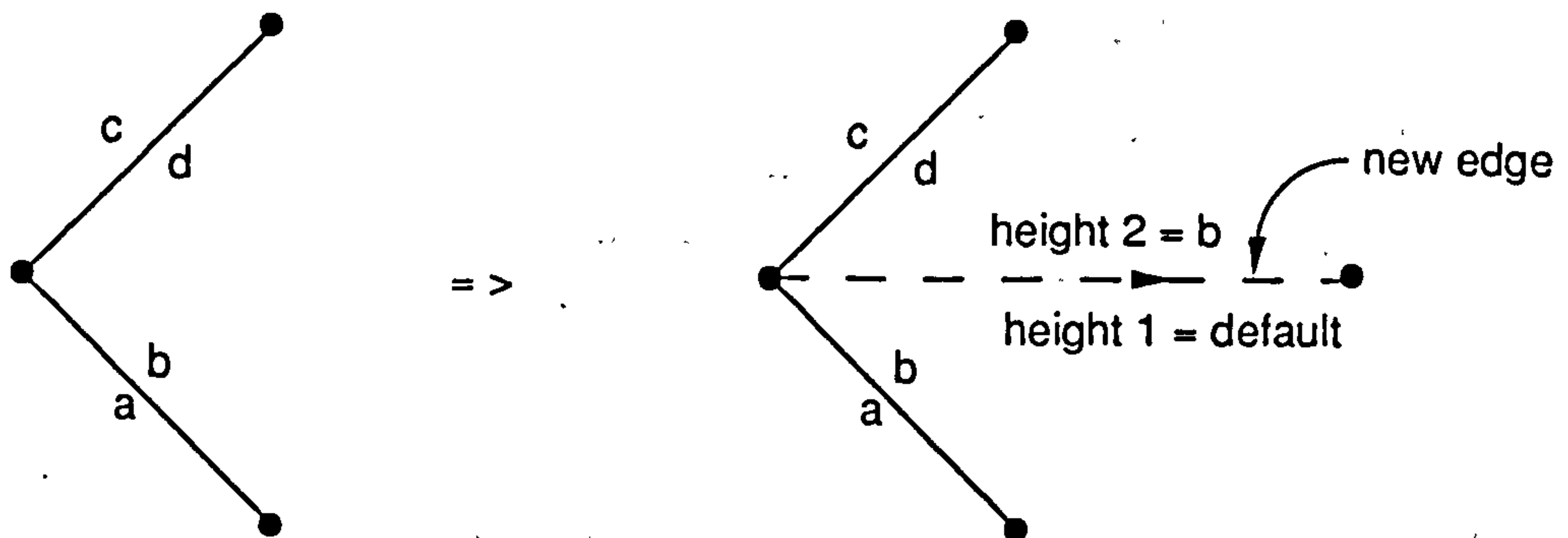
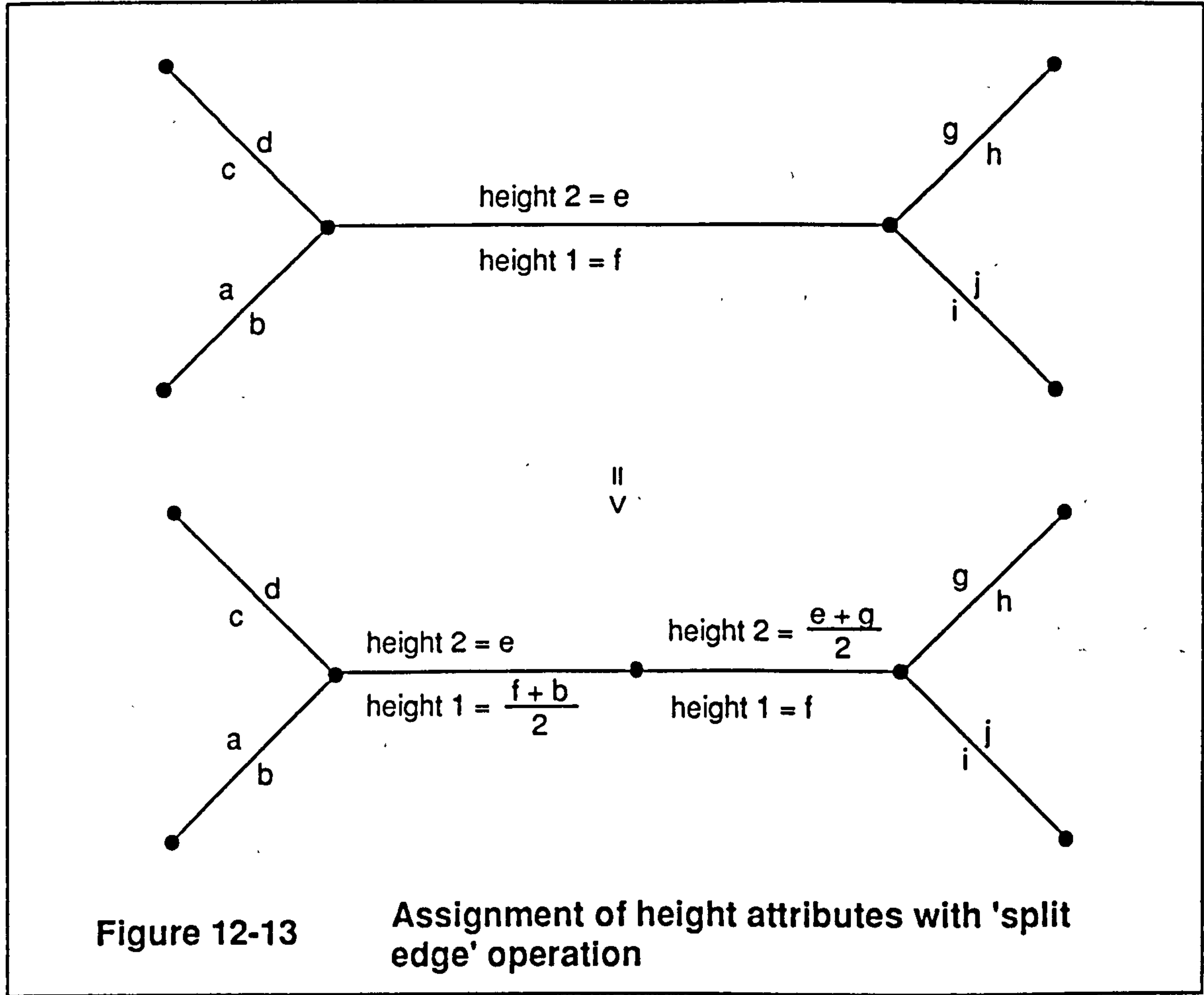
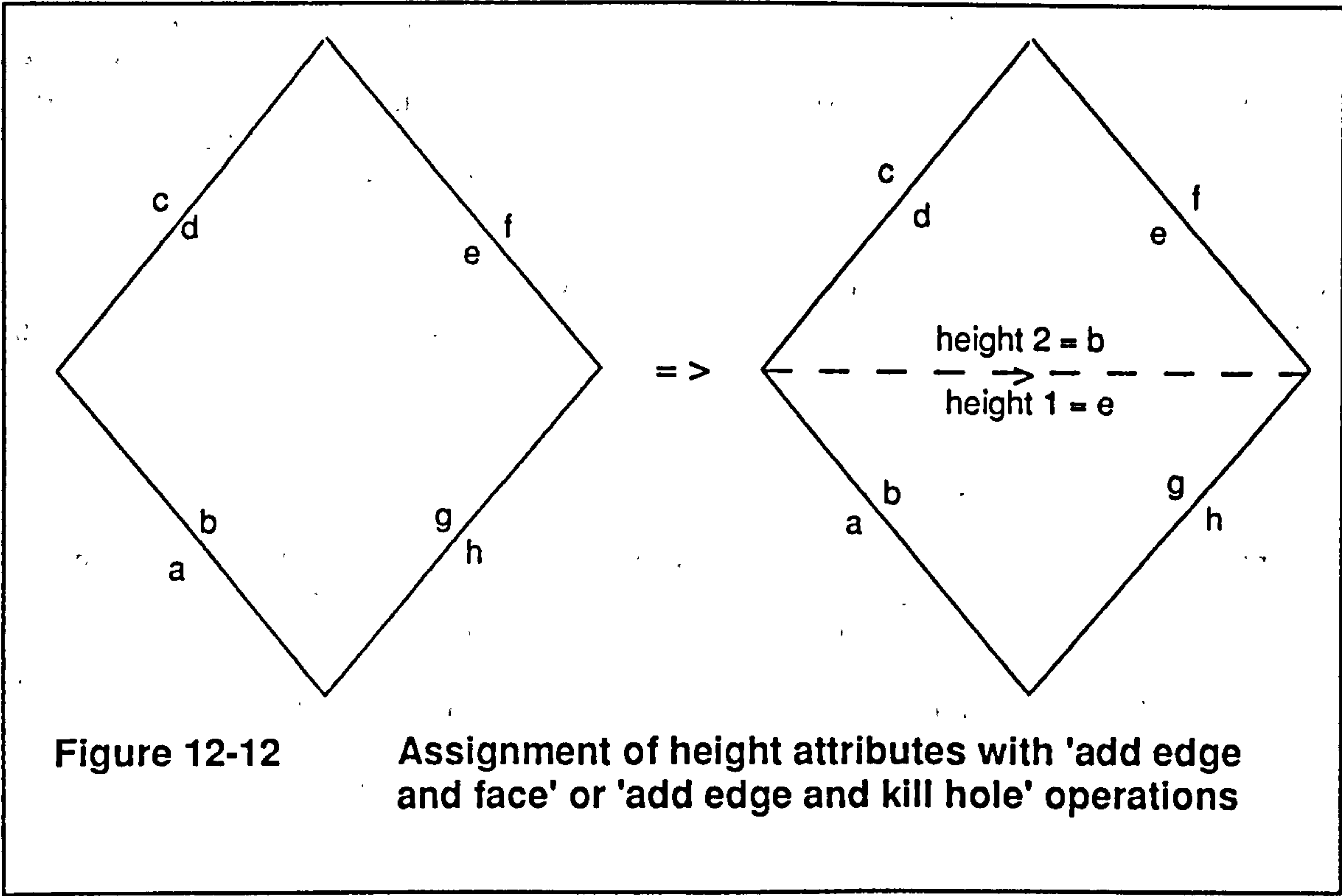


Figure 12-11

Assignment of height attributes with 'add edge and vertex' operation



When a subsequent floor is input, then a point on the digitiser must be mapped to the existing model of the lower floor(s). This is achieved by displaying the graph of the lower floor from which the user selects a vertex point, and digitises its corresponding position. The scale is defined as above. Similarly when an existing floor is to be modified, its own floor graph is displayed from which a vertex point is selected.

A user option allows for this digitiser mapping to be redefined at any time within the input/edit process for a floor. This allows for a drawing on the digitiser to be moved as convenient.

### 12.5.2 Copying between floors

Floors of one building are often very similar, sometimes identical in multi-storey buildings. They are at least similar in outline, if not in their internal layout of the spaces within the floor. Therefore options to copy either all of another floor to the current floor, or just the perimeter of a floor are important. Any unwanted copied walls can always be deleted.

The copy is achieved by considering each space on the floor being copied, and generating the necessary 2D geometry to be input to create the graph of the current floor:

- a) For each edge of the floor face of a space, the adjacent wall construction is considered:

External wall construction: edge is copied

Internal wall construction: edge is copied only if 'all' the floor is being copied, and the current space is the 'left' space of the construction.

- b) The vertices of the edge define the two endpoints that are input into the graph creation process.

- c) If the originating wall construction has category or window lists, these are copied and the new lists assigned to the resulting graph edge. It is assumed that this process is performed before any other walls are input, so that each wall construction generates only one new graph edge, to ensure the above assignment is correct.

### 12.5.3 3D Displays of Building Model

While a floor graph is being created only a 2D view is displayed. Once the 3D building model is generated, more informative views of the model can be displayed. This is very necessary to give the user feedback on the co-ordination between floors, the result of any complex height assignments, and also to show the construction widths resulting from assigned building elements.

Both elevation and perspective views of the 3D model are generated as a wireline or hidden surface display.



### 12.5.3.1 Wireline

To generate a wireline display it is necessary to represent the building model by a set of line segments. So that the construction thicknesses are portrayed, these must represent the position of the offset internal space faces, and the external faces.

The line segments are derived, therefore, by considering the edges of each space. A line segment is generated from each edge where the endpoints are at the offset space vertex points. For each external face of a space, a line segment is also generated from each edge of the face at the true i.e. external vertex points. This results in some duplication of external edges, but this does not affect the display.

Deriving line segments by the above method causes two types of edges to be drawn, which the user may not wish to be included to provide a more understandable view:

a) Space edges that do not exist:

These are present in the model because of the part-evaluated status of the model. They bound 'invisible' faces, those referenced by either 'invisible' wall constructions or spacelink faces. The edges belong to multi-edges whose adjacent constructions are all 'invisible'. Although not true multi-edges, they can be detected by considering 'up' multi-edges, which are adjacent to a set of wall constructions, and 'horizontal' multi-edges which are adjacent to one wall construction and two floor or ceiling constructions. Once a fully evaluated model is created, these edges would no longer exist.

b) Space edges that are adjacent to two coplanar faces:

These can be detected on a space basis by comparing the normals of the two adjacent faces. It is not so straightforward when one of the faces belongs to an 'invisible' construction. The next visible face around the 'multi-edge' is required for the comparison.

Lines representing windows are also generated by considering the edges of 3D lamina of the window, positioned at both the internal offset vertex points, and the original i.e. external points.

### 12.5.3.2 Hidden Surface

A hidden surface display is generated from the faces representing the external envelope of the building model, as the model is only viewed from the outside. These faces are the space faces referenced by the external constructions of the model. A colour is defined for each face, by assigning a colour to the Building Elements, as referenced by category records. The colour is then dependent on the Building Element of the main category of the parent construction of the face. A colour display can then be used to check



the assignment of Building Elements to the external wall and ceiling constructions.

The algorithm to produce the hidden surface display is described in (23). The faces are transformed to a viewing co-ordinate system, with the origin at the viewpoint. A sort is then performed on all front facing faces, such that the display is generated by displaying the faces in order, starting from the back and overpainting with the faces towards the front of the view, nearer the viewpoint. At the same time, a rectangular face for each window of the originating construction is displayed to overpaint part of its parent sorted face.

### 12.5.3.3 Floor Plan

A floor plan can be shown that is more informative than the interactive 2D display used for the input and modification of a floor graph, by considering the wall construction thicknesses. 2D walls are represented by two line segments at this thickness.

The line segments that are drawn are produced from the edges of the floor face of each space, and by considering the adjacent wall construction:

- a) A line segment is drawn using the offset vertex points, if the adjacent wall construction is not 'invisible'.
- b) A line segment is also drawn at the original vertex points of the wall construction i.e. the external line of the wall.

A plan representation of each window is also included in the display, whose width is also determined by its parent wall width.

## 12.6 Interface to Architectural CAD

As indicated in Chapter 2, it is desirable to be able to access building geometric data produced by an architectural CAD system, and from this create the required environmental building model. As discussed in Chapter 4, most CAD software applied to architecture, model a building by an assembly of components, either 2D or 3D. These represent the building structure, using wall and window components, or represent fittings such as furniture. Obviously, it is the components representing the structure that need to be accessed and interpreted.

By using the same operations to create a building model from a 2D floor graph, a possible data transfer process is shown in figure 12-14, and is now discussed.

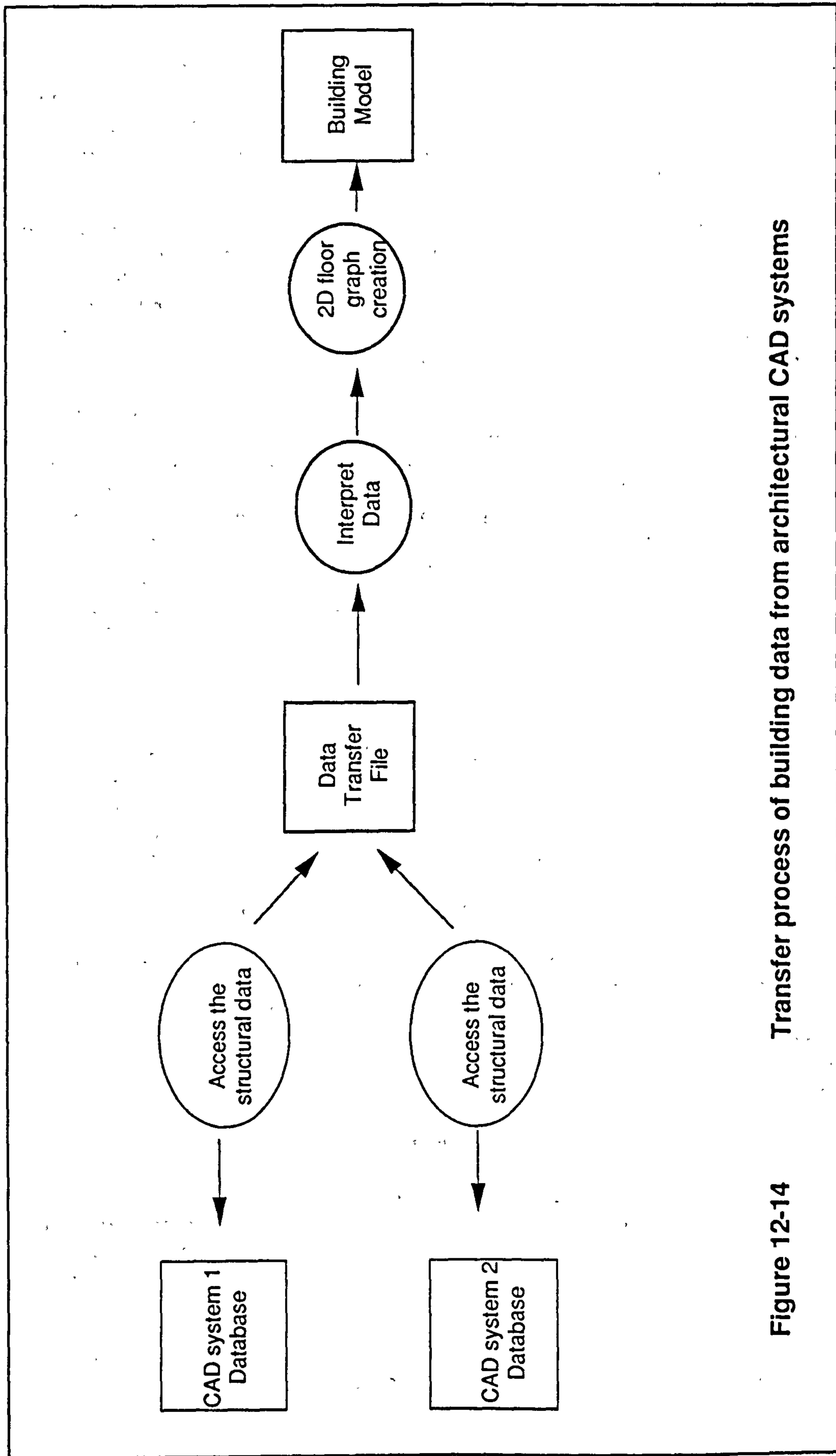


Figure 12-14 Transfer process of building data from architectural CAD systems

### 12.6.1 Transfer File

The transfer file is a character file, that can be read by any computer hardware, in a predefined format, that provides all the available relevant data accessed from a CAD database. The format of this transfer file reflects the input requirements into the 2D floor graph creation process, and must be defined such that it is possible for a number of CAD systems to access their database, and interpret their data accordingly.

It must allow for the definition of walls and windows to be input, and optionally for the 2D graph attributes to be included where available. Data specifying floors and their positions must also be included. This means that a 2D CAD system can generate a data file defining walls and windows in 2D, using default floor to floor heights for the third dimension, or a 3D CAD system can specify walls with heights wherever possible.

There exist a number of data file formats for transferring geometric data at present: IGES and DXF. However these files do not represent structured data that can be interpreted consistently, that is a wall is not always represented in the same way. These file formats could be used if a number of rules were imposed such that it could be guaranteed that the resulting files conformed to a particular structure, i.e. if the requirements described here were adhered to, but that is in effect defining a different file format.

The file format that has been implemented is described in Appendix C.

### 12.6.2 Access of Data from CAD Database

This process is obviously dependent on the format and structure of the CAD database, and a different process is required for each CAD system. It is assumed that it is possible to access the CAD database and extract the instance data for just the structural components, that is walls and windows. This is a reasonable assumption as when modelling a building such structural components are usually grouped together either on the same drawing layer, or by assigning some database type code to the components.

A process to access a GABLE database was developed. As described in Chapter 4, in GABLE the structural entities of a building model: walls, windows/doors, floor and ceiling planes are specifically distinguished from the fittings etc. A 3D model is created, with a building defined as a set of floors. Therefore it is a relatively simple process to access this data to produce a file in the required format.

Some interpretation of data may be necessary in order that all possible data can be transferred. For example, complicated 3D roof structures are modelled in GABLE by a set of roof facets defined by a set of 3D points. Each of these correspond to a space ceiling plane in the environmental building model. These can therefore be transferred by defining the edges of the facets as a set of walls of 'invisible wall construction' type, with the 3D points specified as a 2D point and height.

It is possible that some data which would be of use to input into the building model creation process is not transferred. An example is the connectivity



data in GABLE specifying a set of linked walls. However, it cannot be ensured that all connections are specified, and obviously some CAD systems may not store any. Ideally the file transfer mechanism should be flexible enough to use such data where available, and it is possible that the format could be extended in such a manner.

### 12.6.3 Data Interpretation

This process is in place of the user interface in that it generates data for input into the 2D graph creation process. Therefore, instead of being an interactive process, where the user is provided with feedback, this can be compared to a 'batch' process, that interprets all the data and creates a building model. Obviously once the model is created, the usual user interface can be used to modify and complete the model as necessary. For example to define and allocate building elements, or to assign zone numbers.

The main data item within the transfer file defines a wall for input into the 2D floor graph creation process. The data for each wall is therefore as given in table 12-12.

Data Item	Description
x1, y1	2D co-ordinates of 1st endpoint
x2, y2	2D co-ordinates of 2nd endpoint
type	Corresponds to offset type, and defines the line of the wall
element (optional)	Building element name
leftheight1 (optional)	Height at 1st end of wall, left side
leftheight2 (optional)	Height at 2nd end of wall, left side
rightheight1 (optional)	Height at 1st end of wall, right side
rightheight2 (optional)	Height at 2nd end of wall, right side

Table 12-12: Data Items defining Transferred Wall

Note that some data is optional. By providing two heights at each end, there may be redundant data when walls are joined, but this format simplifies the height specification. The building element name refers to another transferred data item with which the wall width is defined.

It cannot be assumed that the data that is transferred will automatically produce a well-defined planar graph with connected walls. It depends on how the CAD system models and positions walls, in particular where walls abut. In the example shown in figure 12-15, the two connected walls would be defined by endpoints that are not co-incident. Adjustment could be made within the graph creation process, by ensuring that the graph tolerance value was set appropriately. However, using this method does not always achieve the desired result, especially as the walls may be defined in quite a random order, whereby an adjustment of one wall may mean a wall defined later would not join. Therefore assumptions are made with respect to how a wall has been positioned in the CAD system, so constraining the position of

the resulting graph edge, similar to the constraints imposed when a user input points, as discussed earlier.

The assumption is made therefore, that the line of the wall is correct i.e. its direction is fixed and that the specified line of the wall, centre or side, passes through the endpoints. However, it may be necessary to adjust the endpoints to lengthen the line, or choose a different side, so setting a different offset type.

The following process for one input wall achieves this:

1. For each non-parallel graph edge created so far:

- a) the intersection point of the wall line with the edge line is calculated.
- b) the distances of the two edge endpoints from the intersection point are calculated.
- c) the distances of the two wall endpoints from the intersection point are calculated.
- d) these distances are compared with the graph tolerance to determine if either of the wall endpoints is 'at a vertex' of the edge, or is 'on the edge'. See figure 12-16.
- e) these distances are compared with the graph tolerance to determine if either of the edge endpoints is 'on the wall line'. See figure 12-17.

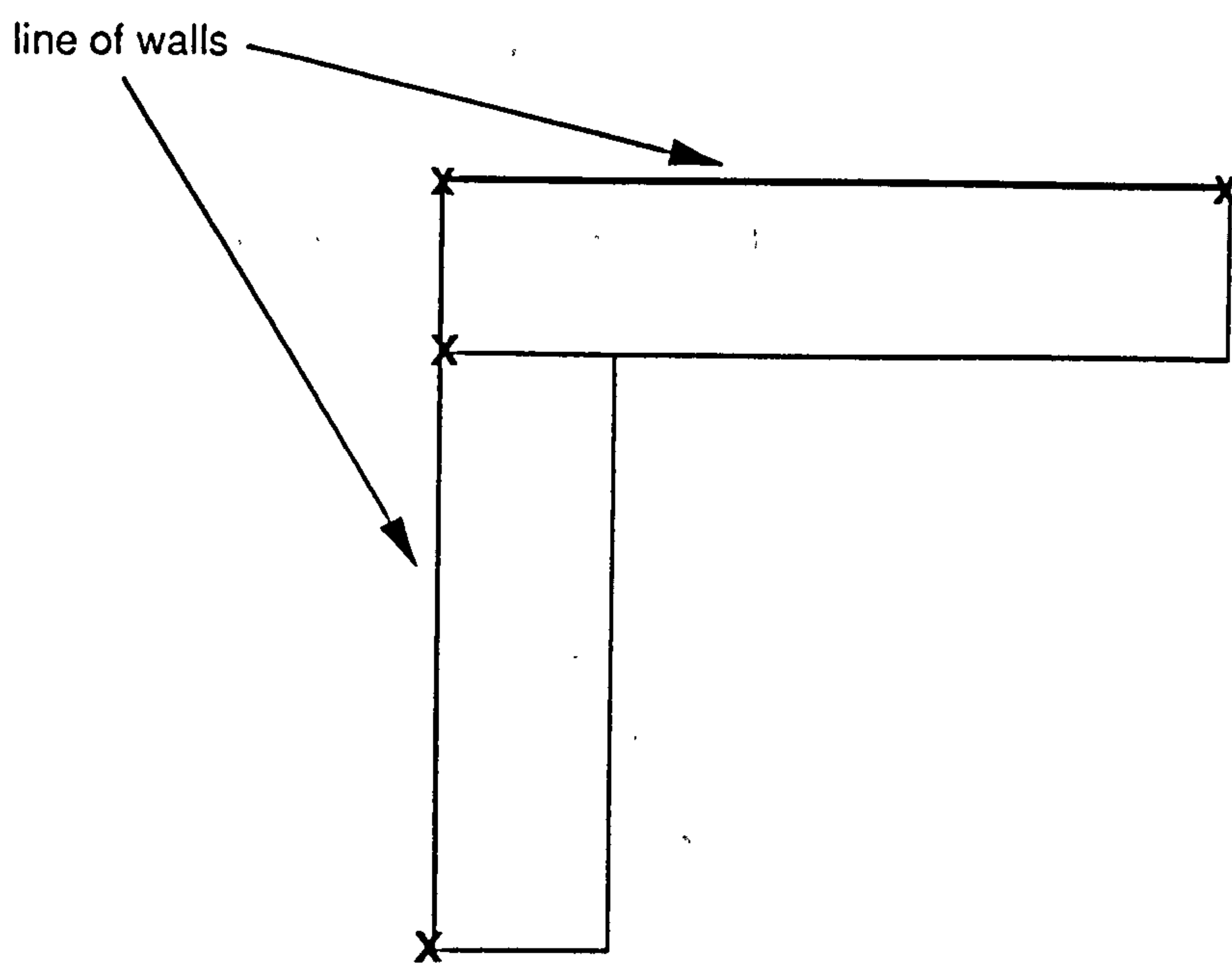
This step produces the position of the endpoints of the wall, with respect to the existing edges and vertices, and also a list of vertices which are 'on the wall line'.

2. It is checked whether any of the existing vertices specified in the above step are 'constraining' vertices, i.e. cannot be adjusted. See figure 12-18.

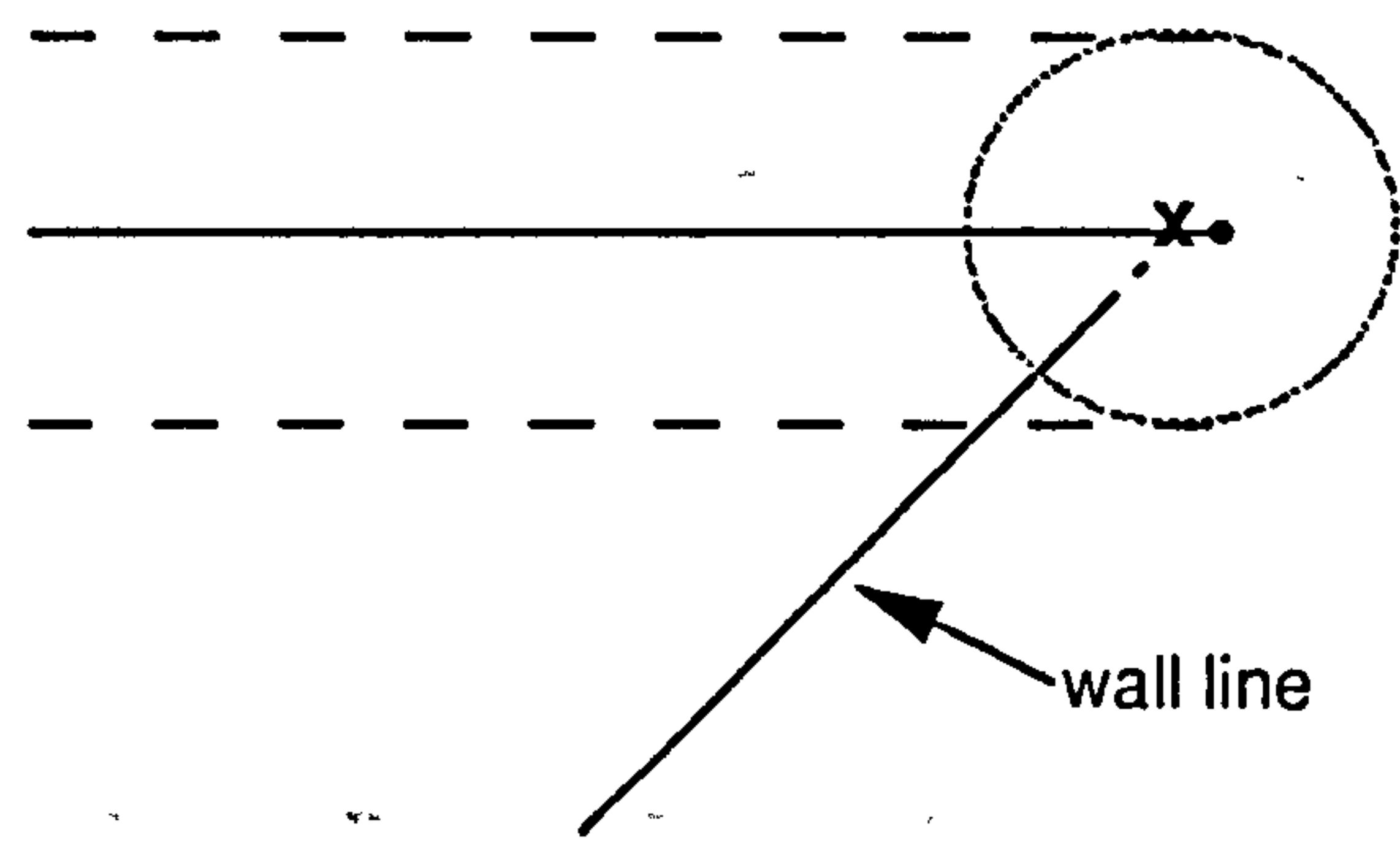
These are vertices which:

- a) have more than one edge or have one edge parallel to the new edge and
- b) are not 'exactly' on the wall line, i.e. are within the graph tolerance distance, but not within a distance that is due to computational error.

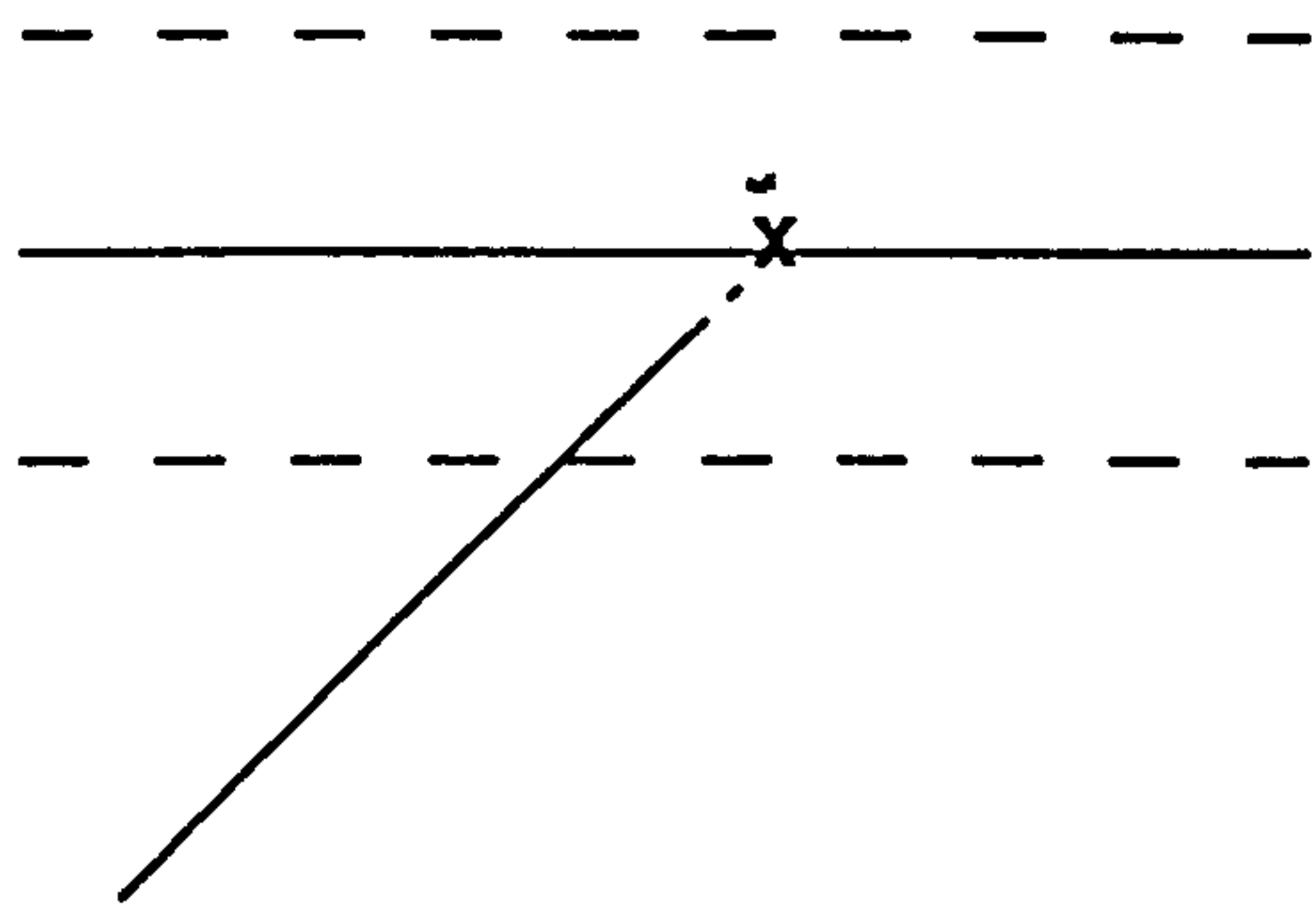




**Figure 12-15.** Example of two connected walls with non-coincident end points

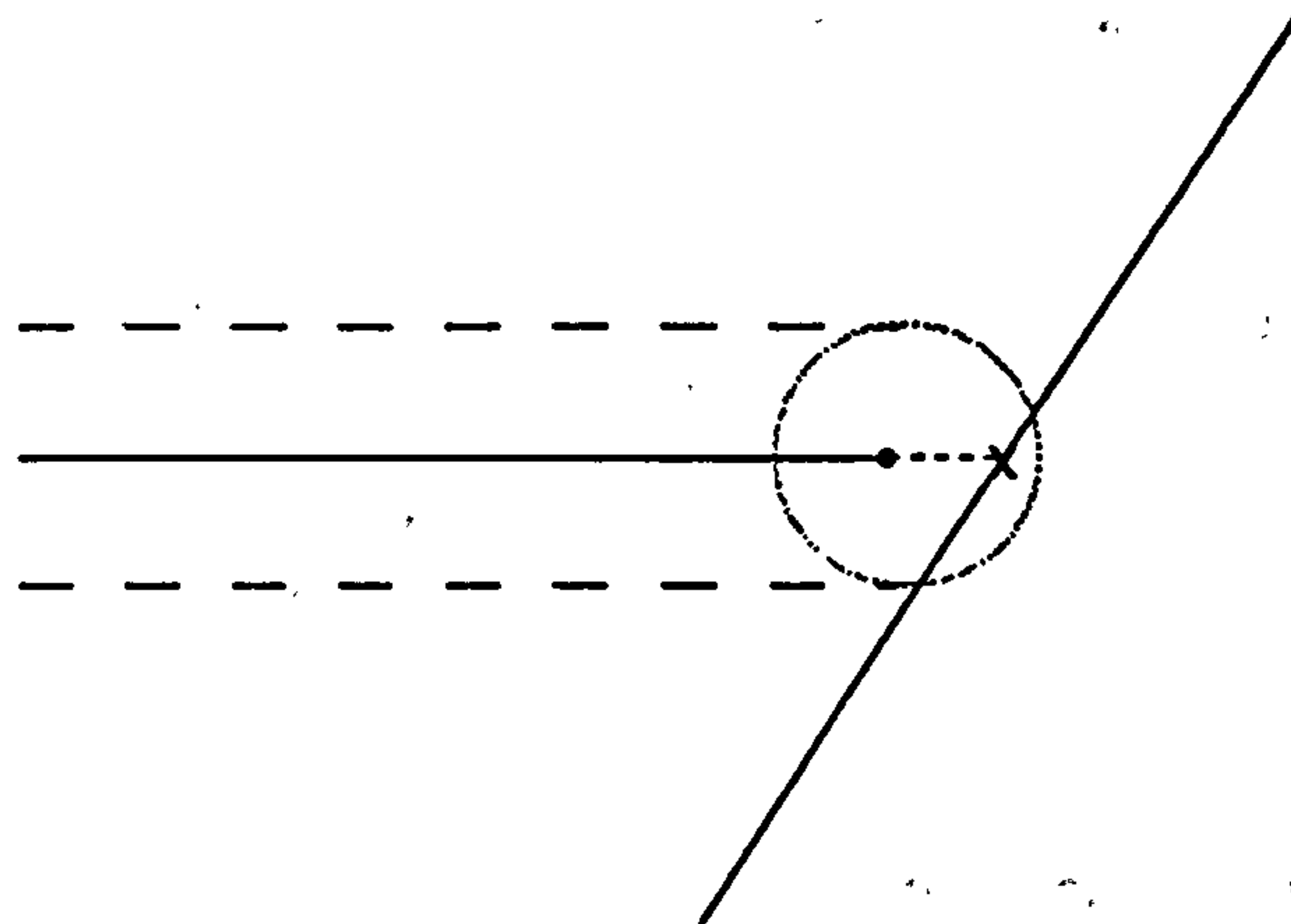


Intersection of wall line with edge is 'at a vertex'

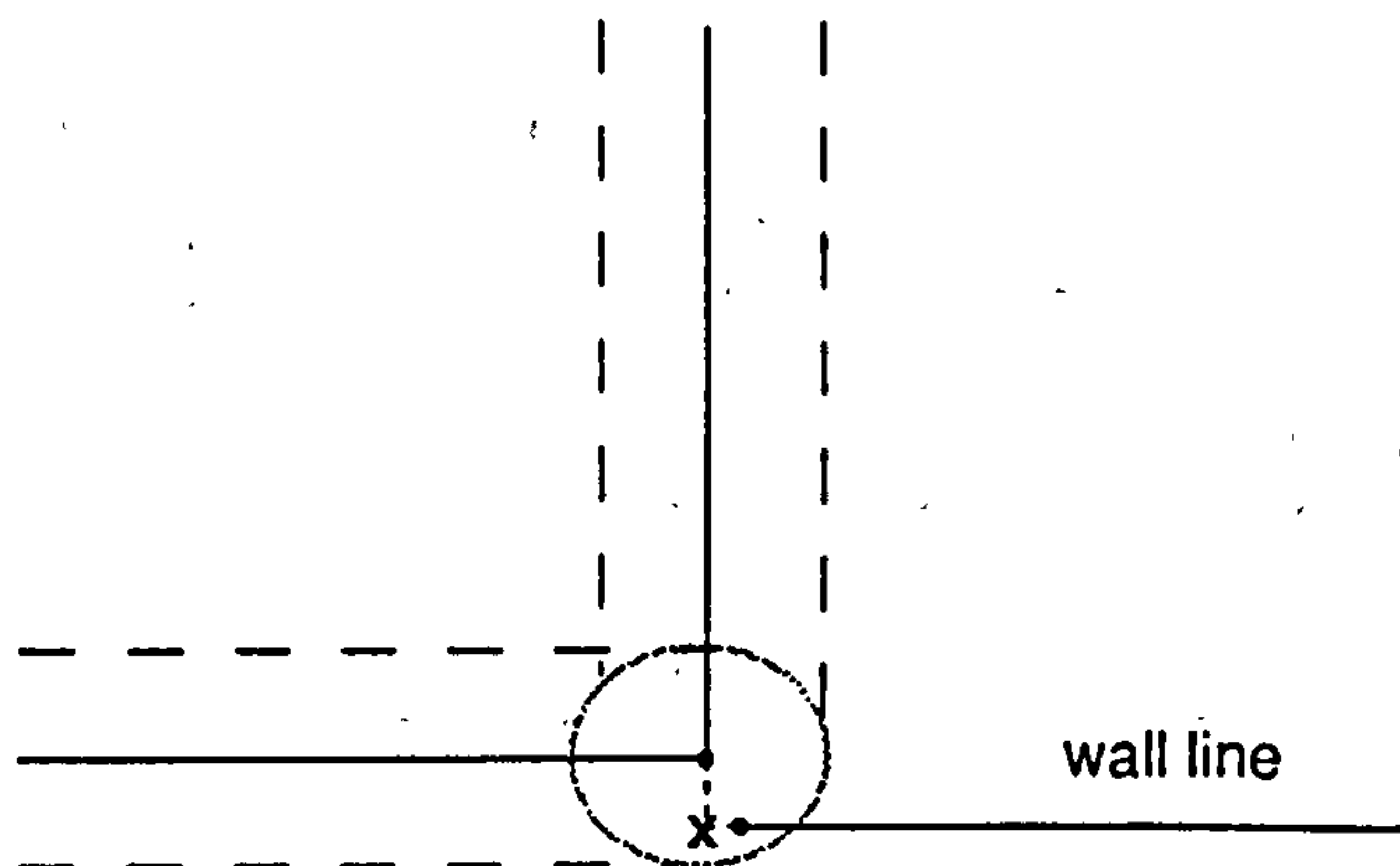


Intersection of point of wall line with edge is 'on the edge'

**Figure 12-16      Using intersection point to determine the position of the wall endpoints**



**Figure 12-17** Using intersection point to determine that a vertex is 'on the wall line'



**Figure 12-18** Example of constraining vertex with two edges

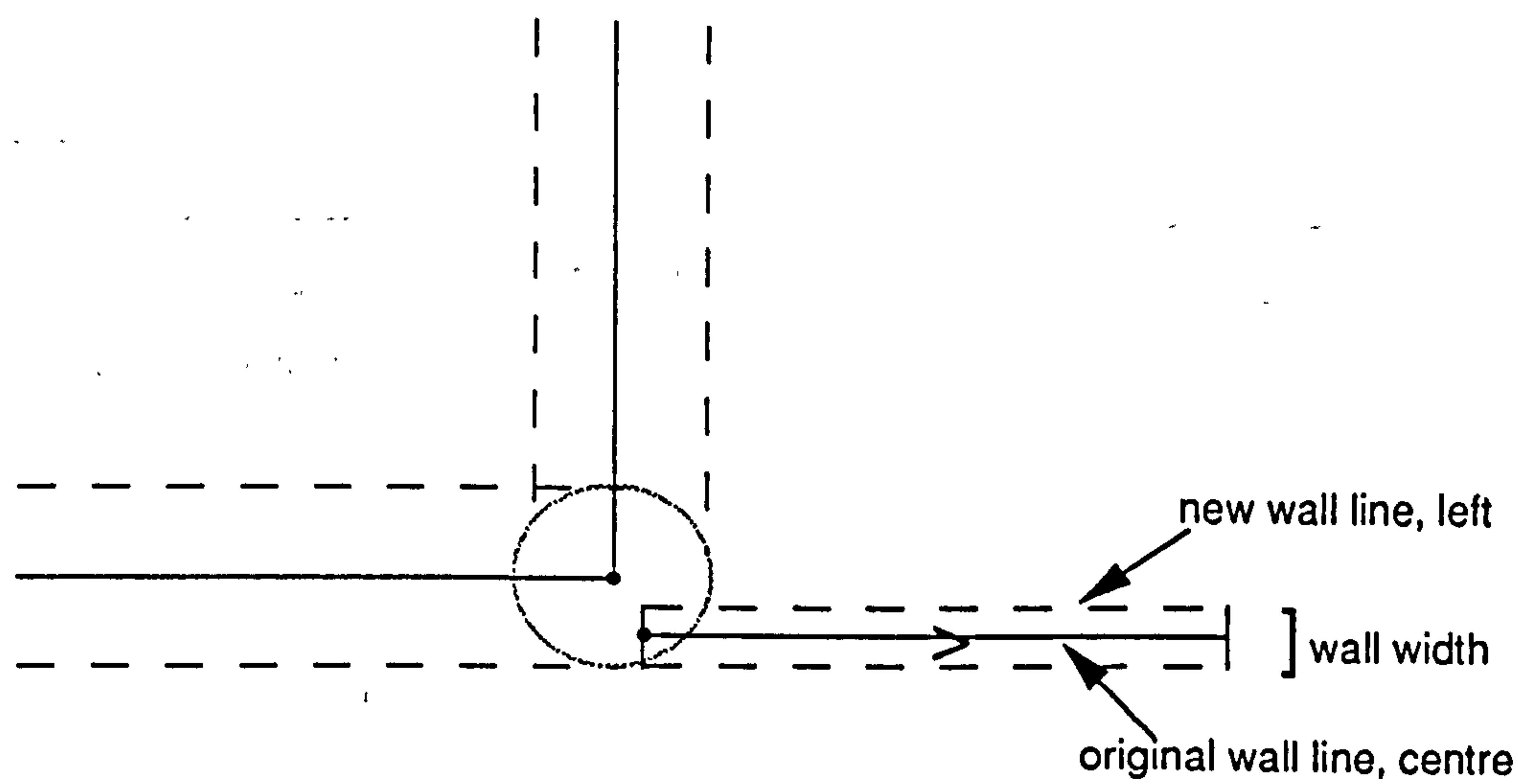
3. If there are any constraining vertices:

- a) The three lines of the wall are calculated, left, right and centre, using the wall width specified by the assigned building element data item.
- b) The nearest line to each of the constraining vertices is found by calculating the perpendicular distance to each.
- c) If there is only one constraining vertex, then the line of the wall is adjusted by the smallest amount possible by using the nearest line to define the offset type. See figure 12-19.
- d) If there are two or more constraining vertices, with the same nearest line, then the line of the wall is adjusted to pass through the first two constraining vertices, i.e the direction of the wall is adjusted by the smallest amount. See figure 12-20. It is necessary to repeat the process so far with the new wall line definition, and also for c) above, as the new wall line may pass close to different vertices.
- e) In other cases it is not possible to fit an exact line. It is not often that there are more than two constraining vertices, although it may occur on a non-orthogonal floor plan. A more frequent case is where there are two constraining vertices to which it is not possible to define the same nearest line, as in figure 12-21. In this case the centre line of the wall is chosen, and the direction of the resulting edge has to be adjusted quite significantly.

4. The result of the above is a wall defined by two endpoints, and an offset type, and a list of non-constraining vertices along the wall line.

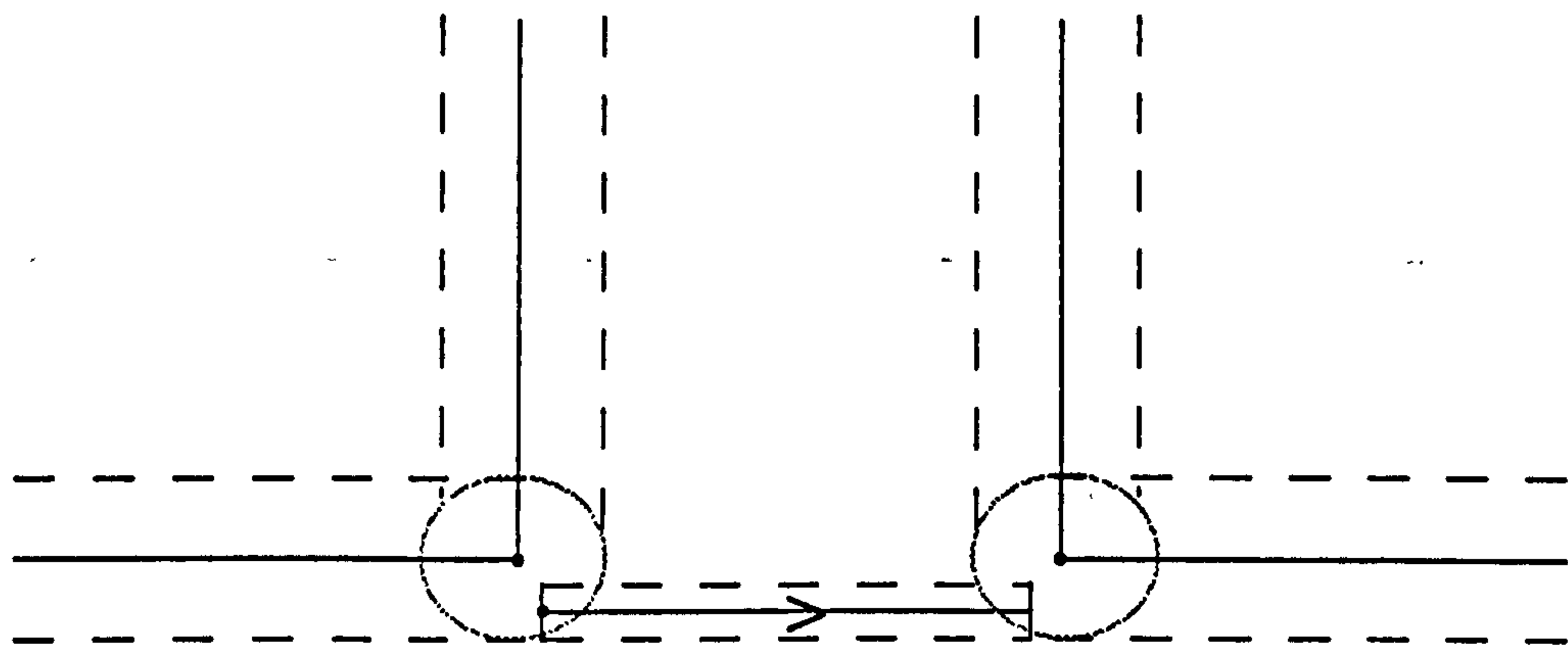
- a) Each non-constraining vertex is adjusted to lie on the line, i.e. the length of its connected edge is adjusted.
- b) The line is input into the 2D graph creation process, with the guarantee that any vertices that are connected to or passed through lie 'exactly' on the line.

An example of a transferred floor plan is included in Appendix C.

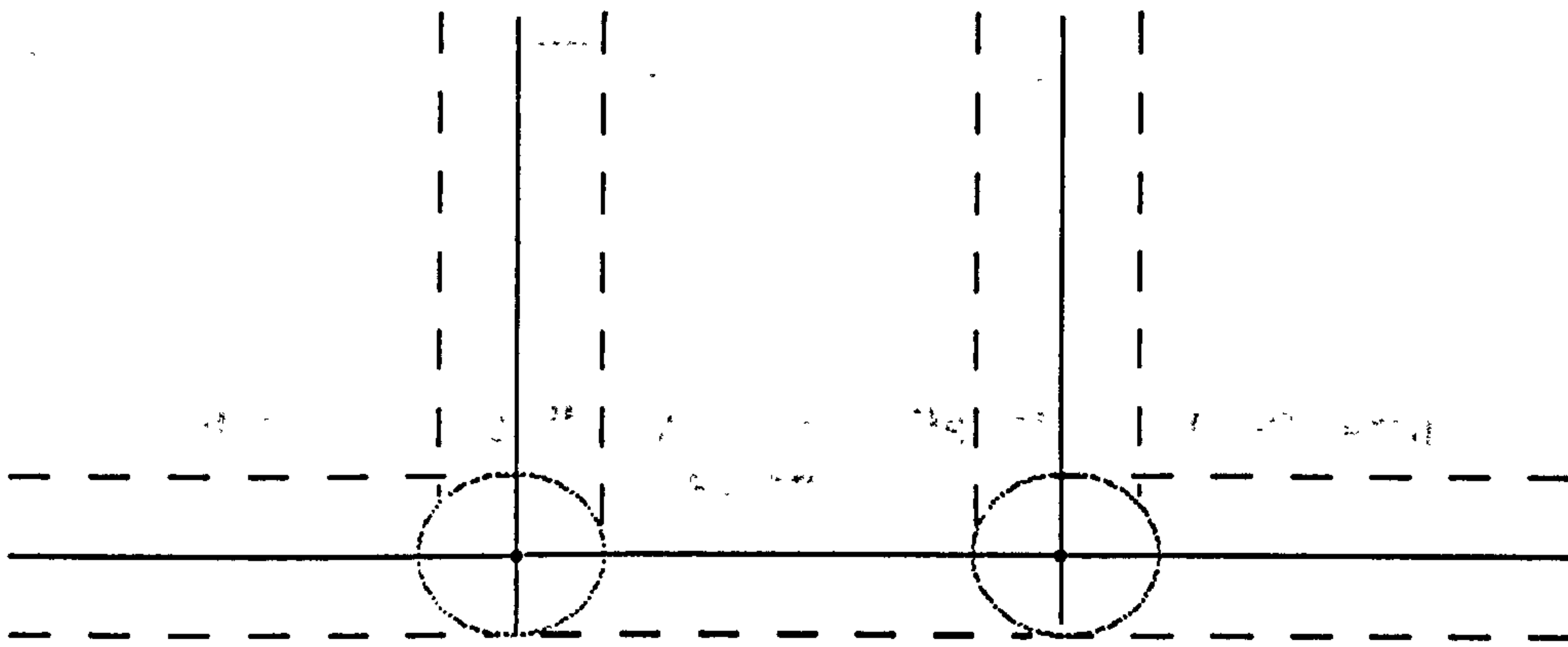


**Figure 12-19**      **Redefinition of wall line to attach to  
constraining vertex**



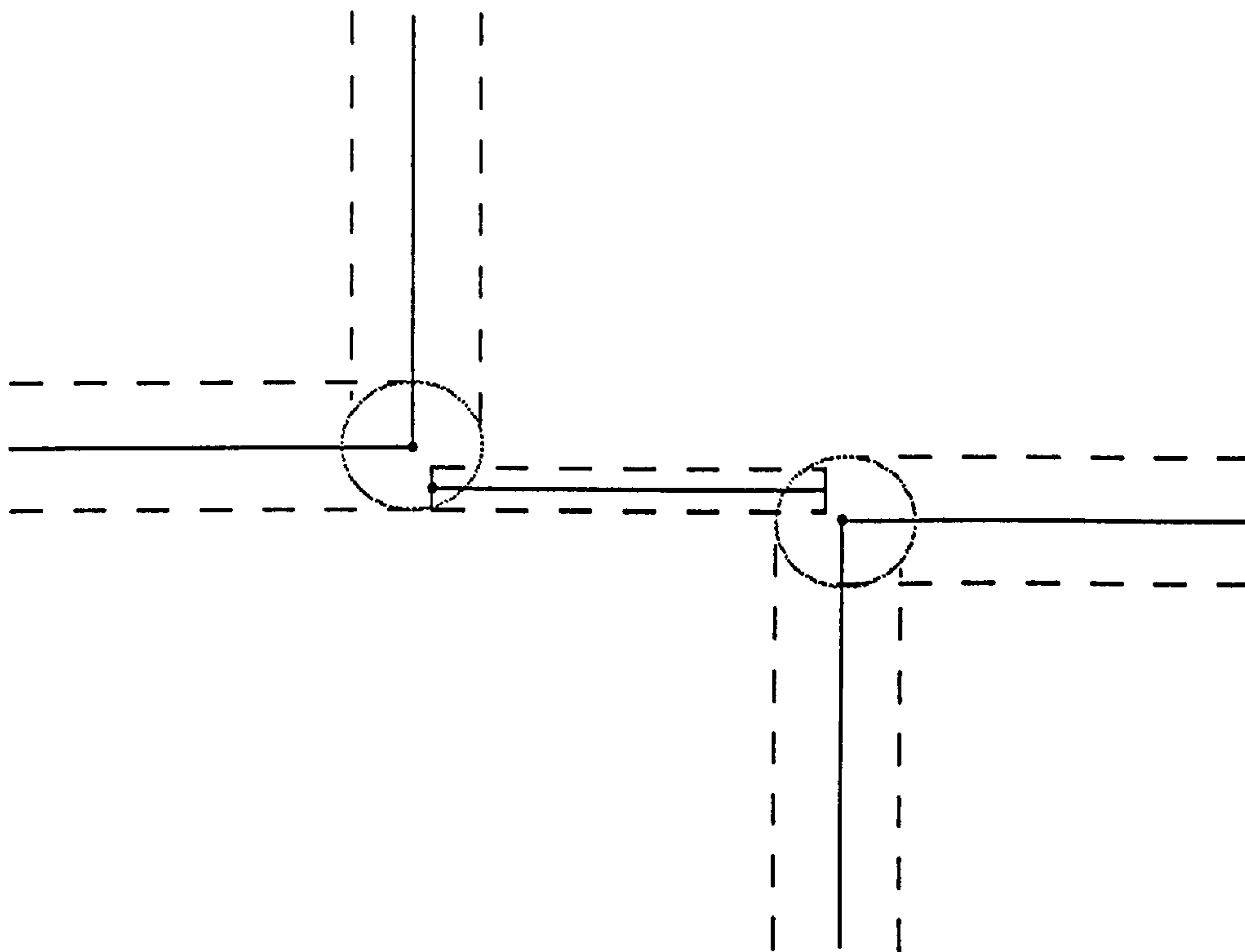


Two constraining vertices, left side is nearest to both



Resulting wall, with direction adjusted by smallest amount

**Figure 12-20**    **Redefinition of wall line to attach to two constraining vertices**



**Figure 12-21**      **Example of two constraining vertices with differing nearest wall lines**



### 13. INTERFACE TO ANALYSIS

Once the fully evaluated building model has been created, as discussed in Chapter 10, it can then be accessed to generate the data as required for any thermal analysis. An interface to TAS has been implemented to generate the data in the form of an intermediate data file, which is then input into the analysis module of TAS.

This is now discussed, and also the generation of shading data, used both for analytical purposes, and for the creation of graphical displays.

#### 13.1 Building Position

Before a building model can be analysed it is necessary to position the building in an environment. This includes defining the orientation of the building, and its latitude position, so that the direction of the sun can be calculated with respect to each surface of the building at any time throughout a year period. Depending on the analysis, it may also be necessary to provide more details on the type of site in which the building is located e.g. on the top of a hill, where the wind speeds may be greater. Options are therefore required to allow the user to define this data.

A user interface option has been implemented to set the orientation of the building and requires an angle to be input. This angle is the orientation of the north vector with respect to the building plan as input. A north 'vector' is then shown on all building plan displays. The default position is a vector 'up' the display, i.e. in the direction of the y axis.

A latitude value can also be input.

#### 13.2 TAS Analysis Data

As described in Chapter 3, TAS requires data on a zone by zone basis, that is, a set of data describing all of the surfaces of each zone. The analysis module of TAS was modified in conjunction with the development of the interface, so for example, it is no longer assumed that a zone is presented by a cuboid, and surfaces with any orientation with respect to the zone are allowed. The transferred data consists of the following items:

- a) General Building data: building name and description
- b) Zone Building Element names
- c) For each zone:
  - i) Volume and floor area
  - ii) Surface data

### 13.2.1 Zone Building Element Names

This is the data as described in 3.2.1, i.e. a set of names, each name being referred to by a number and is referenced by the zone surface data. These names are a slightly different set of names to those in the set of Model Building Element names as described in 11.2, and are generated from the following:

- a) Model Building Element names as referenced by wall construction category records.
- b) Composite Model Building Element names as referenced by the evaluated floor/ceiling constructions.
- c) Window type names, as referenced by each window record.

A construction type from the constructions database is allocated to each of these names.

### 13.2.2 Zone Data

The total volume of a zone, and its total floor area are required. This replaces the dimensions of a cuboidal zone.

The surface data generated from the model differs slightly to that described in Chapter 3, and is summarised below:

- a) Orientation: angle of surface with respect to north. It is no longer necessary to give a building orientation, a zone orientation and relative surface orientations, when the absolute orientation of each surface can easily be calculated.
- b) Slope: as previously defined, except all surfaces have a defined slope, as horizontal surfaces are no longer identified by an orientation label of 'Floor' or 'Ceiling', but by having a slope of zero.
- c) Area: as previously defined.
- d) Building Feature type and linking zone: these types are revised as:
  - i) External: this corresponds to exposed/transparent. Whether a surface is transparent or not is determined by the construction type assigned to the Building Element of the surface.
  - ii) Ground Floor: as before.
  - iii) Dynamic: as before, with linking zone number.



iv) Adiabatic: internal surface, where the adjacent space is not assigned to a zone.

v) Internal partition: internal surface, where the adjacent space is assigned to the same zone.

e) Building Element number: as before, referencing Zone Building Element names above.

N.B. The building model has no concept of a 'shading type', and this data is not generated from the model at present.

### 13.3 Generation of Analysis Data

To create the above data, it is necessary to interrogate the building model, and order the data on a zone basis, rather than according to a set of spaces. Therefore, for each zone, the set of building spaces are searched to find the spaces allocated to the zone.

#### 13.3.1 Zone Volume and Floor Area

The volume of the zone is the total of the volumes of all spaces of the zone. It is relatively straightforward to calculate the volume of a space from a winged-edge data structure with planar faces. The internal volume of the zone is required, and so the internal offset point is used for each face vertex within the calculation.

The internal floor area is required, and is calculated for each space from the floor face, again using the internal offset points, and then summed to give the total area.

#### 13.3.2 Zone Surface Data

To create this data each face of the spaces of the zone is processed. From each face one or more sets of surface data as described above can be determined. This set of data is termed a 'zone face'. A zone face is generated from each category record of the parent construction and from each window record, as placed in the construction.

The data of one zone face may be combined with another in the same zone, if all the characteristics are identical. In this case the areas are summed, so creating the zone surface data. There is not, therefore, a one-to-one correspondence between space faces and zone surfaces. This process is illustrated in figure 13-1.

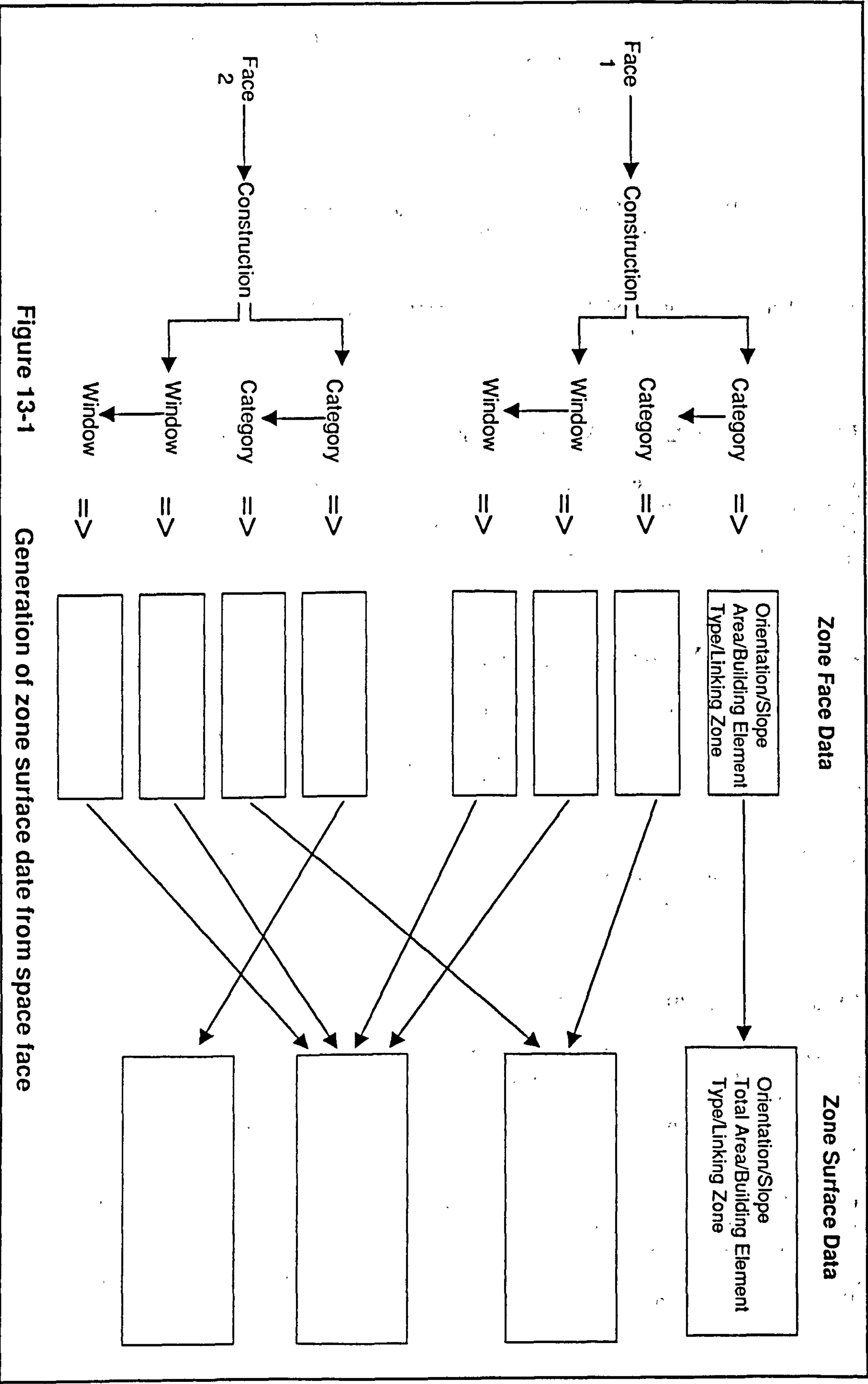


Figure 13-1

Generation of zone surface data from space face

The process generating the zone face data from one face of a space within the zone is now described.

### 1. Orientation and slope:

A normal direction is held within the construction referenced by the face. The direction of the normal is into the space of the first face of the construction. The angles are calculated by considering the normal direction out of the space, so therefore this normal is reversed if it is the first face of the construction being considered. The orientation angle is the difference between the normal vector and the north vector in the xy plane. The slope angle is the angle of inclination of the face.

### 2. Building Feature type and linking zone:

This is determined by considering the construction and the allocation of a zone to any adjacent space.

a) If there is no adjacent space, the type is external. For an external floor construction it is necessary to distinguish between 'External' and 'Ground Floor', and this decision is based purely on the floor number at present: the lowest floor of the building is assumed to be adjacent to the ground.

b) If there is an adjacent space, with no zone number allocated, the type is adiabatic.

c) If there is an adjacent space, with the same zone number, i.e. the current zone being processed, the type is internal partition.

d) If there is an adjacent space, with a different zone number, the type is dynamic, and the zone number is the required linking zone data item.

### 3. Total Area:

The total area of the sets of surface data generated from this face is calculated as being the average of two areas. The areas generated for each side of an internal construction must be equal for the analysis. Therefore the area for both faces of the construction is calculated as being the average of the areas calculated using the internal offset points of the faces. For an external construction, the area is calculated as the average of the internal offset area and the external area, calculated from the original points.

The method by which areas are derived from the building model, where the constructions have an implied thickness, is therefore defined within the interface to analysis process as required by the analysis, and is no longer under the users control. Therefore, the generation of areas from the model is consistent and is not dependent on the user's interpretation. Another analysis method may require different areas to be calculated from the constructions, possibly taking the corner junctions of the space into account.



#### 4. Default Building Element and Orientation:

For constructions with no category records, the default building element, as specified in 11.1, is determined by considering the type of construction: wall or floor/ceiling, whether it is internal or external, and for an external floor construction, whether it is on the lowest floor of the building or not.

As described in 3.2, it is necessary to indicate the orientation of the layers of material, defined within a construction type, as they are applied to a construction. For an external construction, it is assumed that the orientation is as defined in the construction type, i.e. the layers are ordered from inside to outside. For an internal construction, it is necessary to devise a method by which the orientation of the construction type when applied to a construction can be determined. For internal floor/ceiling constructions, a rule can be defined stating that the layers are always ordered from floor layer down to ceiling layer. However for internal wall constructions the orientation needs to be within the users control. Even though most construction types as applied to internal walls have symmetric layers, this cannot be assumed.

The offset types 'left' and 'right' within a wall construction have been used to define this orientation: if the face under consideration corresponds to the offset position of the construction, the construction type is not reversed, otherwise if it corresponds to the face at the original position, the building element is reversed. This is consistent with an external wall construction, where only the offset internal face is considered, and the construction type is as defined. See figure 13-2.

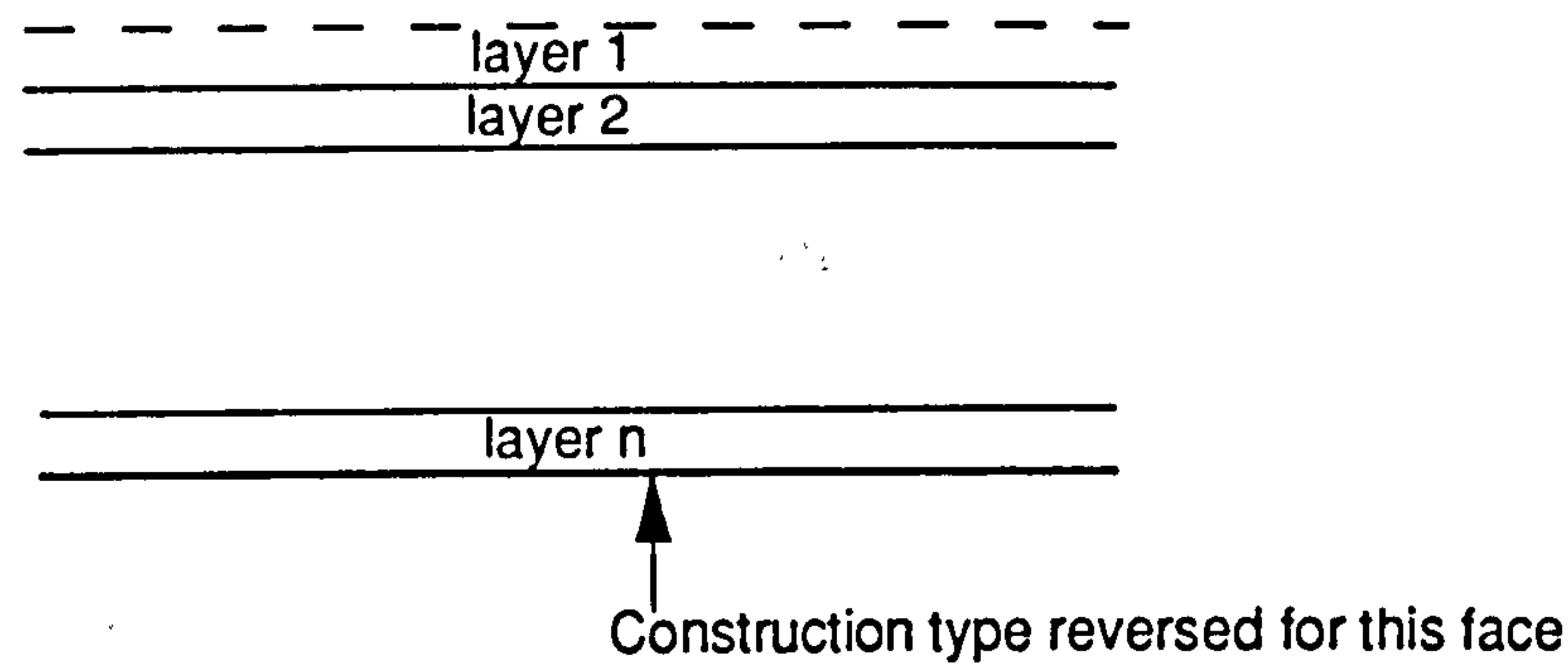
For a 'centre' offset type, the arbitrary decision is made, that the construction type is reversed for the second face of the construction. This ensures that the data is consistent for analysis: there exists two similar surfaces, with the same area, with opposite construction type orientations.

#### 5. Categories and Windows:

Once the above data is determined, one or more zone faces can be generated, depending on the allocation of category records, and windows. If there are none, then just one set of zone face data is generated, using the default building element number.

For each window:

- a) The area of the window is calculated from the 3D lamina.
- b) A zone face is generated using this area, the building element number referring to the window type, that has been included within the list of Zone Building Elements. The orientation, slope, feature type, linking zone and building element orientation are as above. N.B. It is not assumed that a window is placed only within an external construction.



**Figure 13-2**      **Defining orientation of building element from offset type**



The total area of all windows is calculated, and is subtracted from the total area defined above.

If there are no category records, a zone face is generated using the data from above, using the default building element, with the remaining area.

Otherwise, for each category record:

a) If a proportion is defined, i.e. it is not the main category record, an area is calculated as a proportion of the total area minus the windows.

b) A zone face is generated using this area, the number referring to the building element referenced by the category record. The orientation, slope, feature type, linking zone and building element orientation are as above.

The total area of these minor category records is calculated. The remaining area from this total and the window total area is the area to be allocated to the main category, and a zone face is generated using this area, the number referring to the building element referenced by the main category record. The orientation, slope, feature type, linking zone and building element orientation are as above.

Once all faces of all the spaces of the zone have been processed, the result is a set of zone surface data. This is then output to the transfer file, before processing the next zone.

### 13.4. Shading Evaluation

As discussed in Chapter 2, shading calculations using the building geometry are important from two aspects:

a) for the layout of buildings in urban design, by evaluating the shadow patterns resulting from various building designs situated on an existing site.

b) for thermal analysis, the amount of solar radiation on the exposed surfaces of a building must be calculated, where the building may be self-shading, or shaded by other buildings.

This functionality has been implemented, therefore, in the form of options that allow the display of shadows in conjunction with perspective views of the building, and a process to calculate sets of shading factors, corresponding to the zone surfaces, that are then also transferred to the thermal analysis module.

#### 13.4.1 Creation of Site

As indicated above, the shading evaluation usually requires the building to be positioned within a site that includes other buildings. Therefore the

functionality must exist to allow the creation of the geometry of these buildings, and to position them in relation to the building model being analysed. It is obviously only necessary to model the external envelope of these buildings to study the shading effects.

This is achieved by modelling each building as one space, drawing the outline in 2D in relation to the plan of the building model, and assigning heights that determine the sweep operation to create the 3 dimensional representation.

#### 13.4.2 Shading Calculations

To produce a display of shadow patterns, or to calculate the amount of solar radiation on a surface, it is necessary to determine for each external face of the building for a given sun position, whether a shadow is cast onto it by other faces in the scene. The required process is therefore similar to a hidden surface calculation, where a 3D transformation rotates the scene as if it is being viewed from the sun position, and then determines which surfaces, or parts of surfaces can be 'seen'.

The result from such a process for each face is, therefore, one of the following:

- a) the face is completely in the sun
- b) the face is completely in shadow
- c) the face is partially in shadow

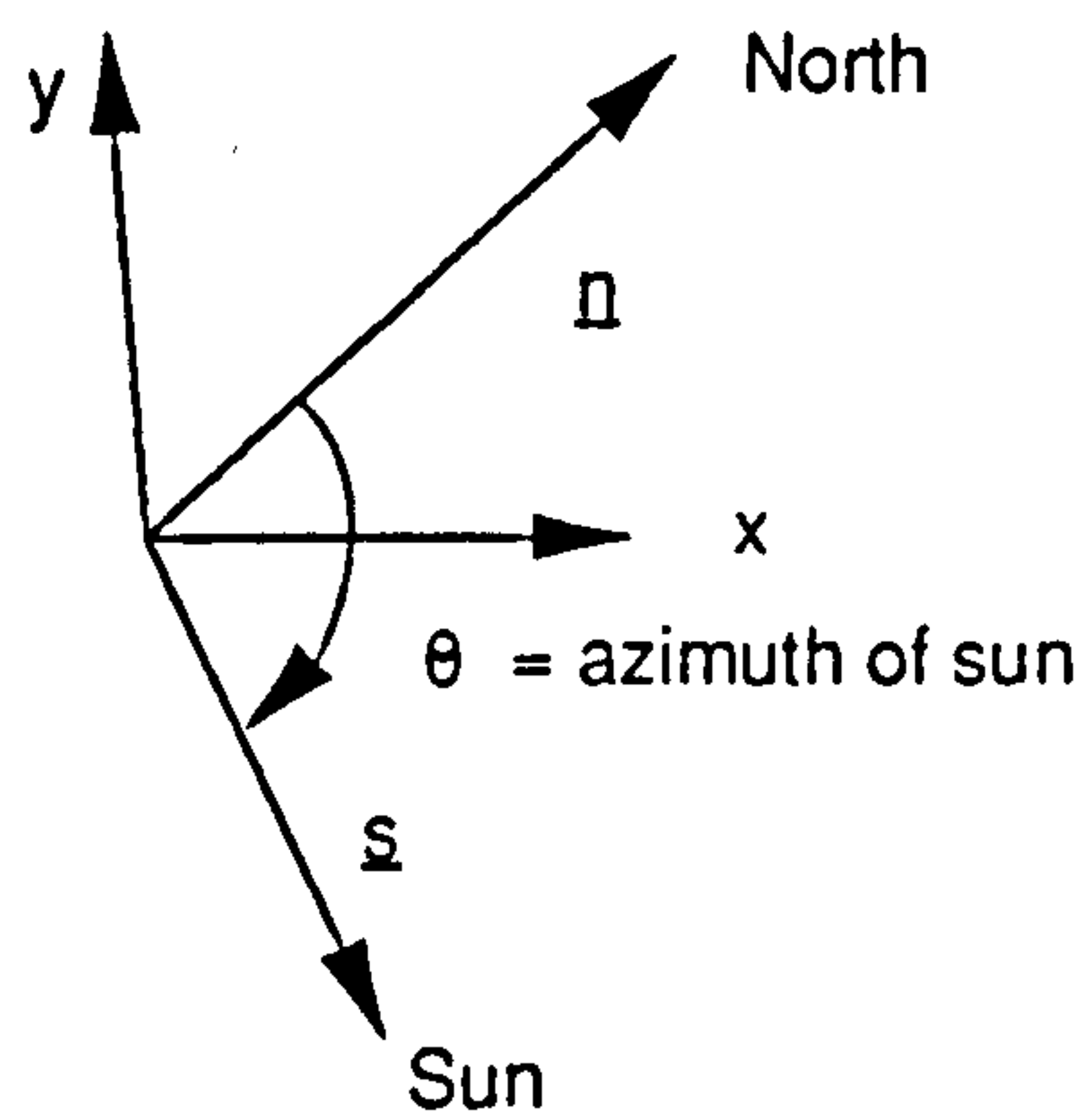
In the latter case, the face must be divided into the areas which are either in shadow or in sun.

The process for a shading calculation for one sun position is therefore as follows:

1. The position of the sun is calculated, and is dependent on the day of the year, the hour of the day, and the latitude at which the model is sited. It is defined as an azimuth angle, with respect to north, and an altitude angle.

A normalised sun direction vector is then calculated with respect to the building model co-ordinate system, which is dependent on the specified north direction. See figure 13-3.

2. The data upon which the shading calculation is to be performed is created. This is similar to the set of faces created for a 3D hidden surface perspective view: it includes all external wall and ceiling faces, but floor faces are always omitted. By comparing the normal of the face with the sun direction, all back facing faces are immediately identified, and not included. The result for these faces is that they are completely in shadow.



$$\underline{n} = a \underline{i} + b \underline{j}$$

$\phi$  = elevation of sun

$$\underline{s} = (a \cos \theta + b \sin \theta) \cos \phi \underline{i} + (b \cos \theta - a \sin \theta) \cos \phi \underline{j} + \sin \phi \underline{k}$$

Figure 13-3

Calculation of sun direction in building model co-ordinate system

3. All points of the data are transformed. A rotation is performed such that the  $z$  axis is in the sun direction, with the positive axis away from the sun. This is not therefore a perspective projection, but a parallel projection, as the sun is an infinite distance away. The faces are then sorted in order of increasing  $z$  co-ordinate.

4. Each face is considered in turn:

All faces in front of the face are projected onto the face in the  $xy$  plane, and the co-incident parts are removed to leave a polygon or set of polygons representing the part of the face in the sun. See figure 13-4.

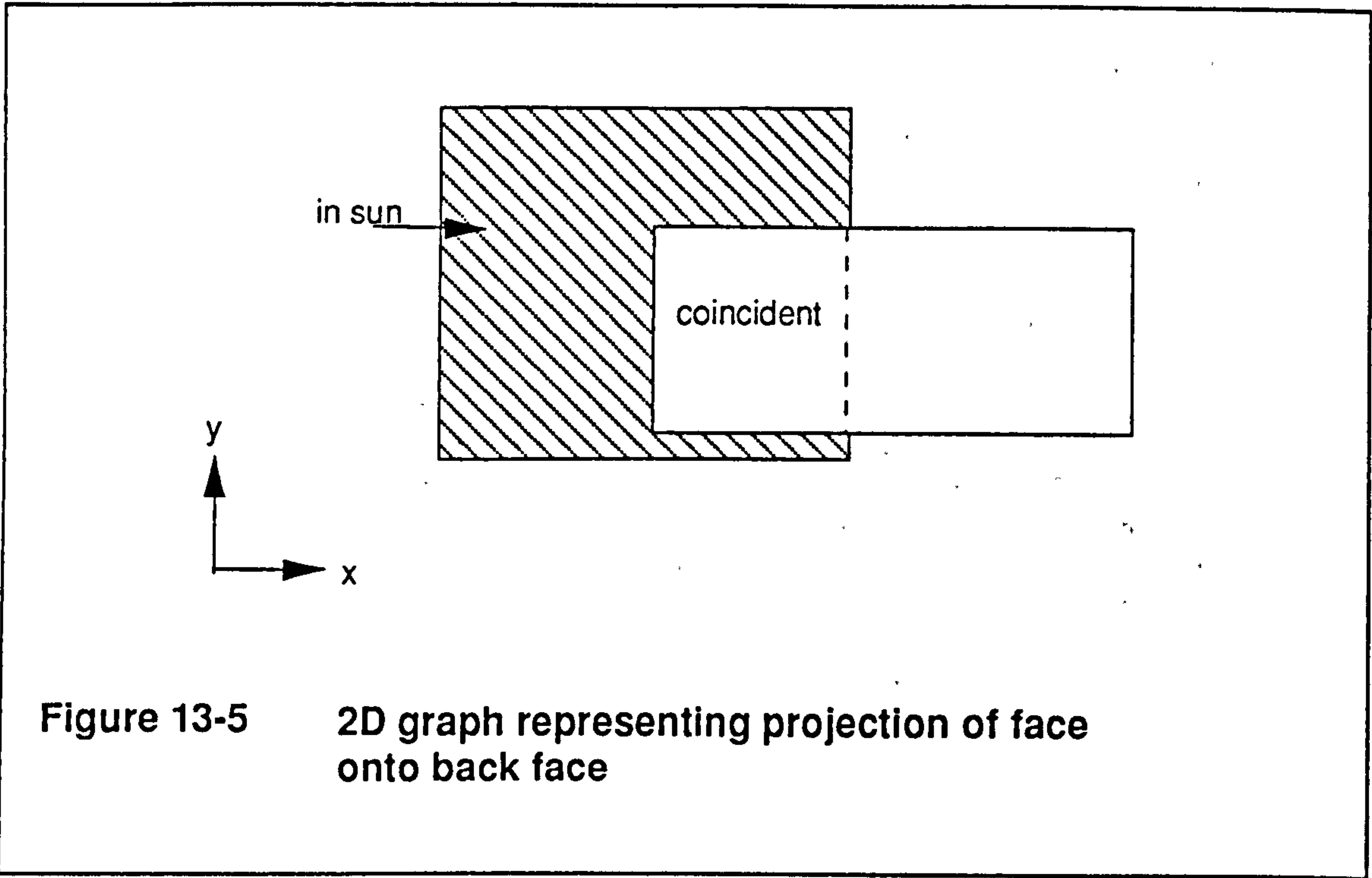
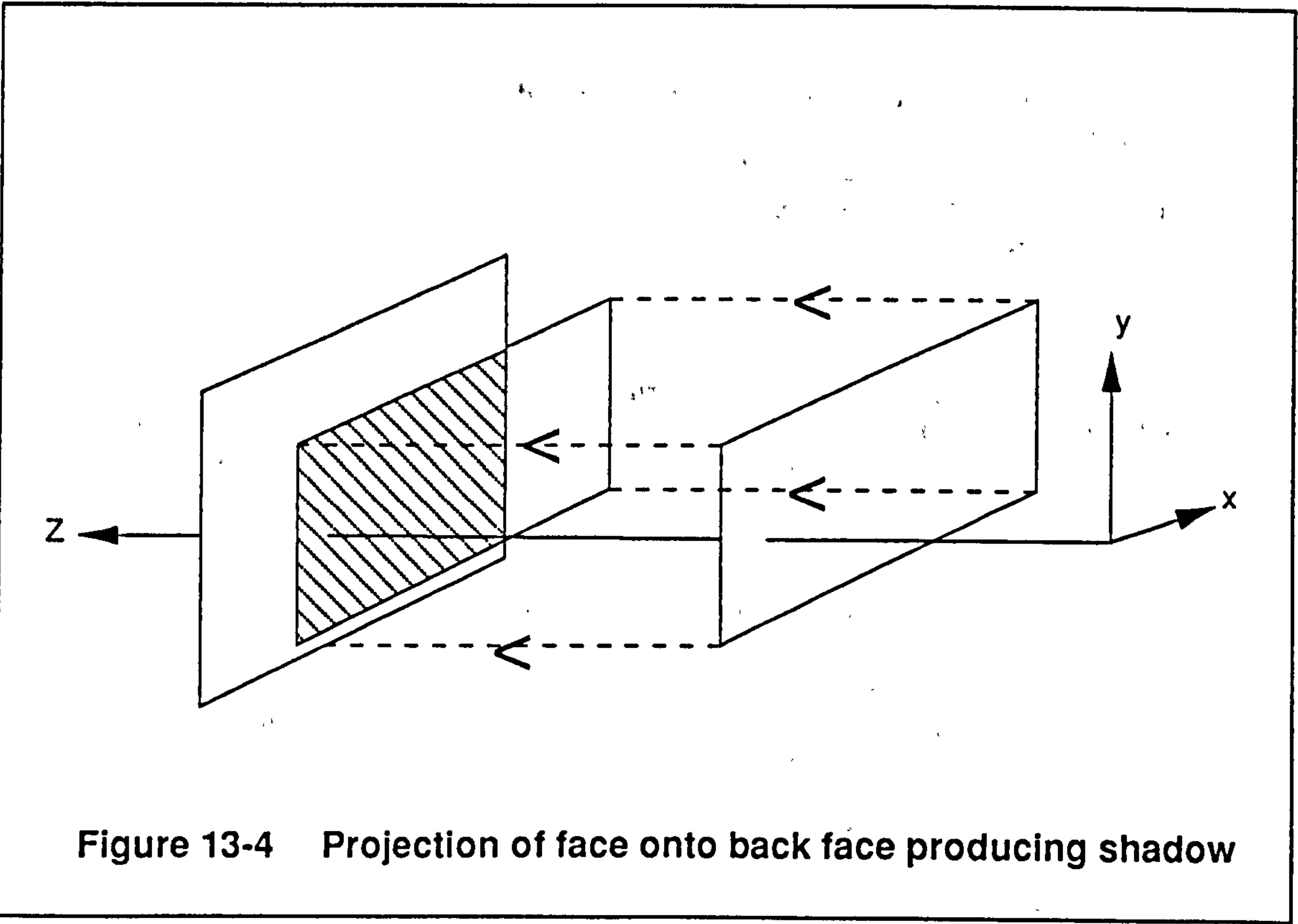
The process used to achieve this is similar to that described in 10.1 for creating floor/ceiling constructions: a 2D graph is created by projecting the face onto the  $xy$  plane, and the edges of each face in front are drawn onto the graph to determine the coincident and non-coincident areas. See figure 13-5. The non-coincident areas of the original face, are the starting 2D graph for the next face. After all faces in front have been considered, the remaining non-coincident areas correspond to the areas of the original face exposed to the sun.

At any time while processing, it may be determined that the face is completely in shade, i.e. only coincident areas are left. There is then no need to continue to process the remaining front faces.

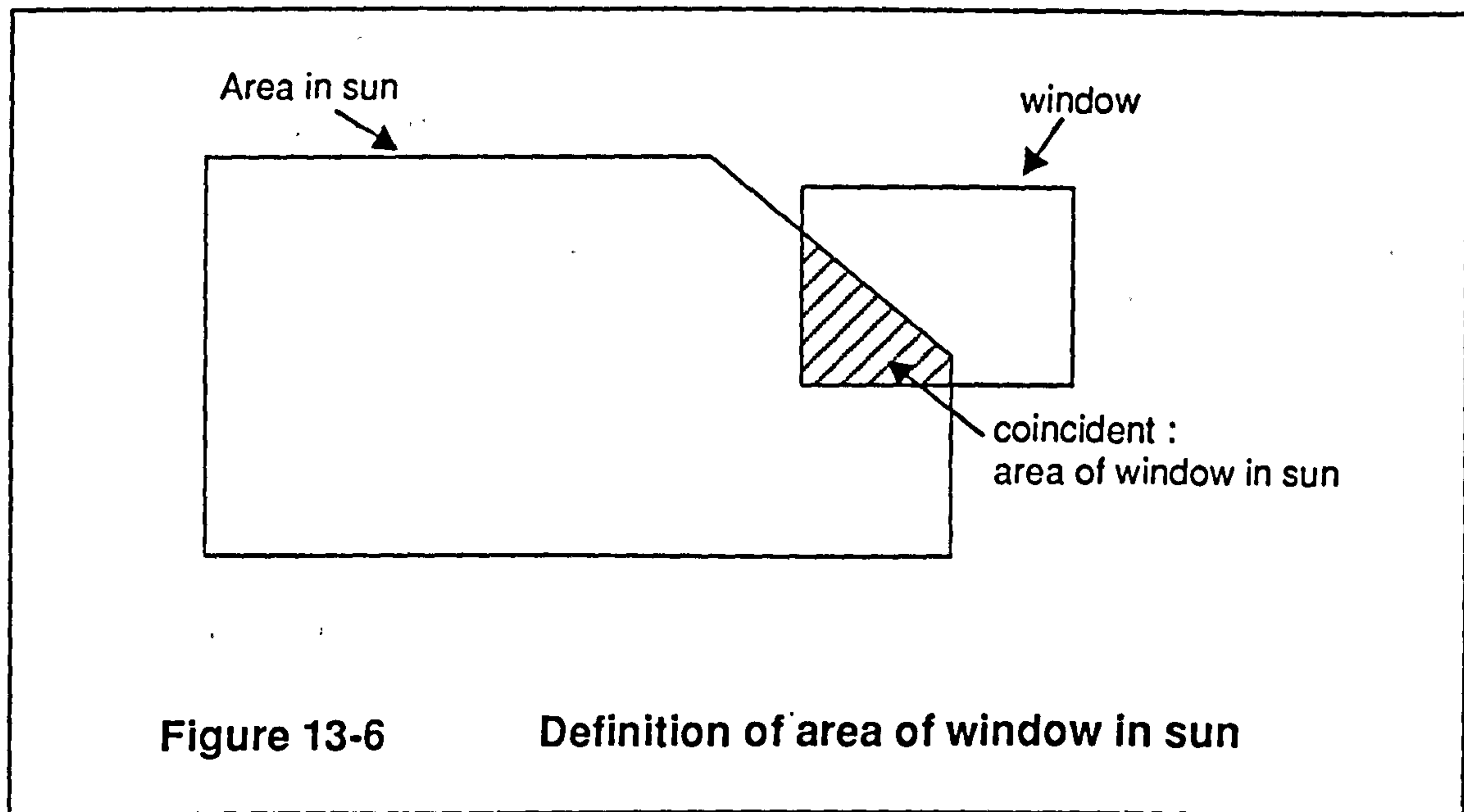
If at the end of the process the face is unchanged i.e. no co-incident areas have been found at all, then the face is completely in the sun.

5. If the face is partially shaded, further calculations are required for each window placed in the face. Given the resulting polygon(s) above, representing the areas in sun, the edges of the window are drawn onto the graph of these polygons. The resulting co-incident areas represent the areas of the window in sun. See figure 13-6. If the window has non-coincident areas, then the window is partially shaded, otherwise it is totally in the sun. If there are no coincident areas at all, the window is totally shaded. The result for each window is therefore also determined.









### 13.4.3 Display of Shadows

When shadows are to be displayed, the result for each face from the above shading calculation is stored temporarily, and referenced either by the original space face, or the window face. The result is therefore either a flag indicating the face is totally in shade or in sun, or is a reference to a winged edge structure representing the resulting set of polygons in the sun.

In order to display these polygons, 3D points must be generated from the 2D points as calculated by the graph process, dependent on the plane of the face in the sun co-ordinate system, that is the z co-ordinate must be calculated. They are then rotated back to the model co-ordinate system. These polygons, representing the shadows at one sun position can therefore be used in several perspective displays, with views taken from different positions.

A hidden surface view is generated as discussed previously. With a defined sun position it is possible to determine a shade of the colour assigned to a face, dependent on the angle of incidence of the sun, i.e. the angle between the sun direction and the normal of the face. Faces directly facing the sun are therefore at full brightness, and faces facing away from the sun are at a level of brightness dependent on the ambient light in the scene. A type of shaded display can be achieved, therefore without including shadows, and is useful to be able to roughly judge the impact of direct sunlight on a building.

A shadow display can be achieved by considering the shading result of each face, sorted according to the perspective view. The faces are displayed starting from the back towards the eyepoint. If the face is totally in shade, it is drawn with the darkest colour shade. If the face is totally in sun, it is drawn with the shade as determined above, dependent on orientation of the face with respect to the sun. If it is partially in the sun, it is drawn as if totally in shade, and then the polygons representing the parts in the sun are drawn on top in the 'sun' shade as above.

The windows of the face are then drawn on top in the same way, that is if partially shaded, first as if totally in shade, and then the parts in sun.

### 13.4.4 Thermal Analysis using Shading Data

To perform a dynamic simulation over a period of time it is necessary to have available for each surface at each timestep the area of the surface that is in direct sunlight i.e. not shaded. Therefore for each zone surface, created from the building model, and transferred to analysis a corresponding set of shading factors is created that indicates the proportion of area of the surface that is in full sun at each required timestep.

For a specific analysis, that is a zone arrangement over a time period, shading calculations as described above are necessary to provide the factors at each hourly timestep. From day to day the position of the sun at each hour of the day differs only slightly, the rate of change dependent on the time of year. Therefore, it is reasonable approximation to perform a shading calculation for the hours of a day, on a weekly basis through the simulation period.

All model faces as specified above are included in the transformation and sort process. However it is necessary to consider the shading of a face only if it is bounding a zoned space. As described above, there is not a one to one relationship between a space face and a zone surface, therefore the shading result of one space face and its windows must be interpreted for the different resulting surface data. That is, the area in full sun for each zone surface is accumulated from the areas corresponding to all its constituent zone face data. Before any areas are calculated, 3D points must again be generated from the 2D points as calculated by the graph process, dependent on the plane of the face in the sun co-ordinate system. However, it is not necessary to transform back into the building model co-ordinate system.

Therefore, given the shading result of a space face, and its windows, the process to accumulate the areas is as follows:

1. Face is totally in shade:

no action.

2. Face is totally in sun:

The area as specified in each dependent zone face is added to the respective zone surface sun area.

3. Face partially in sun:

- a) For each window:

Window totally in shade - no action

Window totally in sun - area specified in related zone face is added to zone surface sun area

Window partially in sun - calculate areas of polygon(s) in sun and add to relevant zone surface sun area.

The total area of all the windows of the face in the sun can then be calculated.

- b) Calculate the areas of the polygon(s) in sun of the face, and subtract the window sun area. The remaining area is divided between any category records.

As the shading polygons are created from the external face of a construction, and the area stored in the zone surface is an average of the internal and external faces, the sun area must be scaled down by the same factor. This will ensure that the total of the surface in the sun is not greater than the zone surface. The factor is:

$$\frac{\text{average area} - \text{total area of windows}}{\text{external area} - \text{total area of windows}}$$

Therefore for each sub-category, the resulting area added to the zone surface sun area is:

sun area x percentage x factor.

The remaining sun area is then attributed to the zone surface area of the main category.

Finally, once all faces bounding zoned spaces have been processed, a shading factor is calculated for each zone surface from its area and sun area. The set of factors, that is for one timestep, are then output to a transfer file before repeating the process for the next timestep.



## 14. CONCLUSION

This thesis has discussed the development of a CAD tool for use in the environmental design of buildings. The following have been achieved:

1. The conceptual building model as implied by the input data to thermal analysis describing the building form has been defined. A data structure has been derived to represent this unambiguous model, and checked for sufficiency in terms of its topological entities. This models a building as a set of spaces, each represented by a winged-edge data structure, and sets of entities, corresponding to the structural building components, representing the coincidence of vertices, edges and faces of the adjacent spaces.

2. A modelling method has been implemented to create this representation. The winged-edge representation of a space is created with a modified solid-modelling technique, by sweeping a 2 dimensional profile, the multiplicity of the other entities being determined by a 2 dimensional planar graph representation of a building floor.

3. The creation of a 2 dimensional planar graph from purely geometrical data has been achieved, where the input data defined by a user represents walls. The importance of defining a graph tolerance was determined, such that the contiguous 2 dimensional plane could be divided into the discrete faces of a planar graph.

4. The use of a winged-edge data structure for two purposes was established, to represent both a 3 dimensional space and a 2 dimensional planar graph, so requiring the specification of additional entities associated with basic topological entities to define this usage. The functions performing the Euler operations to create and modify these entities were specified to allow for this dual usage also.

5. To enable the modelling of more complex 3 dimensional building forms, it was necessary to modify the basic modelling method. Taking into account also the need for the iterative creation and modification of the building model by the user, the requirement for a part-evaluated building model was established. In this model a space is represented by a set of space units with identical topology, the fully evaluated model only being created when analysis is required.

However, a drawback with the modelling method was identified, when modelling buildings with spaces that extend over more than one floor of the building. The restriction is imposed that the building must be created in order of the lowest floor to the highest, with any extended spaces being defined before the higher floors to which it extends are input.



6. The representation of attributes as assigned to the spaces and structural entities completes the building model data structure as required for analysis, such that the internal environment properties are associated with each space, and materials are defined for each structural component. The importance of allocating material definitions by default was emphasised, and this is implemented dependent on the type of component, and its position with reference to the external envelope of the building.

7. An attempt has been made to model the true building geometry, by incorporating the construction thickness into the conceptual model, within which zero thickness is assumed. This allows better visualisation of the building model, but more importantly means that the specification of the required geometrical data, e.g. areas, that is exported to the thermal analysis module, can be determined by the exporting software, rather than by the user, removing discrepancies in interpretation.

8. Rectangular windows were incorporated into the building model representation, a window instance being associated with either a wall or a roof construction, and referencing a list of window types.

9. A user interface was implemented such that a user could define a building model from existing drawings, or sketch in a new design. It was necessary to resolve conflicting constraints on the positioning of walls within the software when creating the planar graph from the user defined geometry. These confictions arose between the user interface options to define the 2 dimensional geometry and the process to create the planar graph where adjustment is necessary to ensure it is well-defined.

10. To define the 3 dimensional geometry of the building and its properties, the user interacts with and assigns attributes to the graph entities. It was necessary to consider the resulting attributes for each Euler operation when adding to the graph or modifying it.

11. An interface to architectural CAD has been implemented such that 2 dimensional data and any 3 dimensional data is used to create the planar graph representation. Assumptions are made concerning the transferred floor representation such that this imposes constraints on the data input into the floor graph creation process. The software attempts to produce a 'best fit' of the data to a floor graph representation.

To achieve this interface, an intermediate file format has been defined to transfer data from any CAD system used for architectural design. More recently there has been effort towards defining the method and format by which a building definition could be transferred between any computer systems (24). The use of such a format could be relevant here.

12. 3D visualisation of the building model has been implemented to enable the model to be checked and the design communicated to others. Displays

incorporating shadows for any sun position can be produced, allowing solar analysis of the building design also.

13. Interfacing software has been developed to access the building model, and produce a data file containing data in the form required for the thermal analysis module of TAS. This orders data on a zone basis, according to the allocated zone attribute to each space of the model. Analysis of different zone arrangements can therefore be achieved easily and quickly.

It is feasible that an interface to other analysis systems could be developed by accessing the building representation and creating data in the input format as required by the analysis module.

14. To enable the effects of shading to be included in the thermal analysis, shading factors can be calculated for each zone surface and transferred also to the analysis module. The factors are calculated on a weekly basis, for each hour of the day.

The implementation of the above has produced a CAD tool that has been successfully employed in modelling a wide range of building forms, before analysis with TAS.

There are a number of possible improvements to the current functionality to be investigated which would extend the usage of the modelling system further. These are itemised below:

1. As indicated above, the model creation method can be restrictive. This could be improved with further modelling facilities, to allow a space to be deleted completely from the representation, making any adjacent internal constructions external, or by detecting the intersection of extended spaces with upper floors, to produce the effect of punching a hole through an upper floor.

2. The creation of complex 3 dimensional geometry can be difficult. The assignment of height as an attribute to the 2 dimensional floor graph means that a 3 dimensional display is not generated until the spaces and constructions are generated, that is once the input and modification of the floor is complete. The user could be assisted by a 3 dimensional display of the floor using the heights assigned so far. Sectional views of the resulting building model through several floors would also be of value in understanding the interaction between floors.

3. Only rectangular window shapes are modelled. Any 2 dimensional shape could be used to represent a window placed in a wall or roof plane. In addition further geometrical entities could be associated with a window definition to model shading features. The topological representation would not be effected that positions the window shape within a wall or roof, the shape defining the area of glazing for analysis. However additional 3

dimensional planes (as in a surface representation) could be used in shading calculations. This functionality could be extended to allow the positioning of any 3 dimensional plane data around or attached to the building representation purely for shading purposes.

4. Using the representation of a space and the windows placed in the constructions adjacent to the space, it would be relatively straightforward to generate internal sun patterns on the faces of the space to enable visualisation of such patterns, and for inclusion within analysis, so assisting further in the environmental design of individual spaces.

5. An assumption is made with regard to the position of the ground: only the lowest floor is adjacent to the ground for analysis purposes. To allow for basements and sloping sites where spaces on different floors may be adjacent to the ground, a definition of a ground surface in relation to the building model is necessary. This would also enable better visualisation of the site.

6. The interface to analysis is one way at present, in that the results of the analysis are shown with respect to the zone surfaces. However, as discussed previously, there is not a one-to-one relationship with the faces of the building space, and this can cause problems in interpretation. This could be improved if the results of the analysis were accessed through the displayed building model, that is by selecting a space face, the results associated with the face could be displayed. Similarly for spaces with respect to the results for each zone.

7. Finally, it is possible that this building representation could be used for other environmental analyses, such as daylight and lighting analysis, by allowing the association of different attributes with the entities of the model, for example surface characteristics, and also the positioning of entities such as light fittings, with which attributes could be associated by referencing an appropriate applications database.



**REFERENCES**

1. DAY, B. Computation of the dynamic thermal performance of buildings, CAD Vol 14, Number 1, Jan 82, pp 49-54.
2. CLARKE, J.A. Computer applications in the design of energy conscious buildings, CAD Vol 14, Number 1, Jan 82, pp 3-9.
3. BOWMAN, N.  
LOMAS, K. Does dynamic simulation work ? Building Services, March 1986, pp 30-31.
4. BLOOMFIELD, D.P. The influence of the user on the results obtained from thermal simulation programs, CIB Fifth International Symposium on the use of computers for environmental engineering related to buildings, July 1986.
5. SONDEREGGER, R.C. Thermal Modelling of Buildings as a Design Tool - Clima 2000, Copenhagen 1985.
6. ROGERS, R.  
NALL D.  
GREENBURG, D. Computer graphics input methods for building energy analysis, Computer Aided Design, Volume 9, Number 3, July 1977.
7. PITTMAN, J.  
GREENBURG, D. An interactive graphics environment for architectural energy simulation, Computer Graphics, Volume 16, Number 3, July 1982.
8. PECKHAM, R.J. Shading evaluations with general three-dimensional models, Computer Aided Design, Volume 17, Number 7, September 1985.
9. TAS User Manual 1985.
10. ESP: A Building and Plant Energy Simulation System, Version 5, Release 3, March 1986.
11. AISH, R. Building Modelling: the key to integrated construction CAD, CIB fifth international symposium on the use of computers for environmental engineering related to buildings, July 1986.
12. EASTMAN, C.M.  
PREISS, K. A review of solid shape modelling based on integrity verification, Computer Aided Design, Volume 16, Number 2, 1984, pp 66-80.

13. GIBLIN, P.G.                      Graphs, surfaces and topology. Chapman and Hall, London 1977.
  
14. BAUMGART, B                      A polyhedron representation for computer vision, AFIPS Conference proceedings, NCC Vol 44, 1975, pp 589-596.
  
15. EASTMAN, C.M.  
    WEILER, K.                      Geometric Modelling using the Euler Operators. Institute of Physical Planning, Carnegie-Mellon University, Research Report No. 78, February 1979.
  
16. WEILER, K.                      Edge-based data structures for Solid Modelling in Curved-Surface Environments. IEEE Computer Graphics And Applications, January 1985, Volume 5, No. 1.
  
17. BRAID, I.C.  
    HILLYARD, R.C.  
    STROUD, I.A.                      Stepwise construction of polyhedra in geometric modelling. C.A.D. Group Document No. 100, University of Cambridge Computer Laboratory, October 1978.
  
18. PAVEY, S.G.                      Euler Operations in a Simple Solid Modeller, MSc Thesis, Cranfield Institute of Technology, Sept 1984.
  
19. MANTYLA, M.                      A Note on the Modelling Space of Euler Operators, Computer Vision, Graphics and Image Processing 26, pp 45-60, 1984.
  
20. SHAPE DATA LTD FOR  
    CAMI-INC                      Geometric Modelling Project, An interface between geometric modellers and application programs, Vol III, Fortran Implementation. 12-80-GM-04, December 1980.
  
21. MITCHELL, W.J.                      Computer Aided Architectural Design, Van Nostrand Reinhold, 1977.
  
22. MARCH, L.  
    STEADMAN, P.                      The Geometry of the Environment, Methuen and Co., 1974.
  
23. FOLEY, J.D.  
    VAN DAM, A.                      Fundamentals of Interactive Computer Graphics, Addison Wesley, 1982.
  
24. GIELINGH, W.                      General Reference Model for AEC Product Data, ISO TC184/SC4/WG1, Document 3.2.2, May 1987.



### ACKNOWLEDGEMENTS

I would like to express my gratitude to my tutor Mr. M Pratt, and to Amazon Computers for the sponsorship for this PhD award. I am indebted to my colleagues at Amazon Computers for the many discussions regarding the functionality of this software tool, and for their encouragement in developing it, in particular, Brian King, Harry Mellor, Alan Jones and Francis Vaughan.

I would like to thank my illustrators, Susan Ringsell and Daphne Watts for the production of the figures within the text, and also Jayne Singh for her assistance and patience in checking, correcting and collating this manuscript.

I am also grateful to Dr. W. Batty, for the use of the computing facilities within the Applied Energy Department, Cranfield and his assistance in producing the colour displays.

Finally, my thanks are to my relatives and friends for their unending support and encouragement while writing this thesis.



**APPENDIX A****DATA RECORDS OF UNEVALUATED BUILDING MODEL****BUILDING**

Field Name	Description
space	1st space of building

**SPACE**

Field Name	Description
nextspace	Next space in list for building
lowerfloor	No. of lowest floor of range of space
upperfloor	No. of highest floor of range of space
floorface	Floor face of space unit of lowest floor
ceilingface	Ceiling face of space unit of highest floor
spacelink	Space link, null if range is one floor only
floorcat	First of linked list of categories for floor face
ceilingcat	First of linked list of categories for ceiling face
zone	zone number

**SPACELINK**

Field Name	Description
space	Next construction in list for building
floorface	Floor face of space unit
ceilingface	Ceiling face of lower space unit, co-incident with floorface
spacelink	Space unit, null if last spacelink in floor range

CONSTRUCTION

Field Name	Description
nextcon	Next construction in list for building
lface	'Left' face of construction.
rface	'Right' face of construction.
normal	3 components of normal direction of left face.
offsettype	Type indicating positions of offset faces: 'centre', 'left' or 'right'
catlist	First of linked list of categories for construction
windlist	First of linked list of window instances placed in construction

NODE INSTANCE

Field Name	Description
point	3D point of node
vertex	Corresponding vertex of winged-edge structure.
offset	3D offset point of vertex
nextnode	Next node instance in list for node

CATEGORY

Field Name	Description
buildelem	Reference to Building Element
percent	Percentage of total area of construction
nextcat	Next category for construction

WINDOW

Field Name	Description
wintype	Reference to Window Type
point1	First 2D position point
point2	Second 2D position point
face	First face of window construction





## APPENDIX B

### IMPLEMENTATION OF USER INTERFACE FOR BUILDING MODELLING SYSTEM

The system was developed on an Apollo DN3000 workstation, for colour displays of 1024 x 800 resolution. The graphical interface was created by using the 2D-GMR graphics library as supplied with Apollo workstations. This library provides windowing and viewport facilities for 2D graphics represented by a segment display list, providing the required functionality for interactive graphical operations. Displays with either 4 colour planes or 8 colour planes can be used, enabling 16 or 256 colours, respectively, to be displayed at one time. The shaded perspective views adapt to the number of available colours, dependent on the number of different colours in the view, and by varying the number of shades of one colour, from bright to dark.

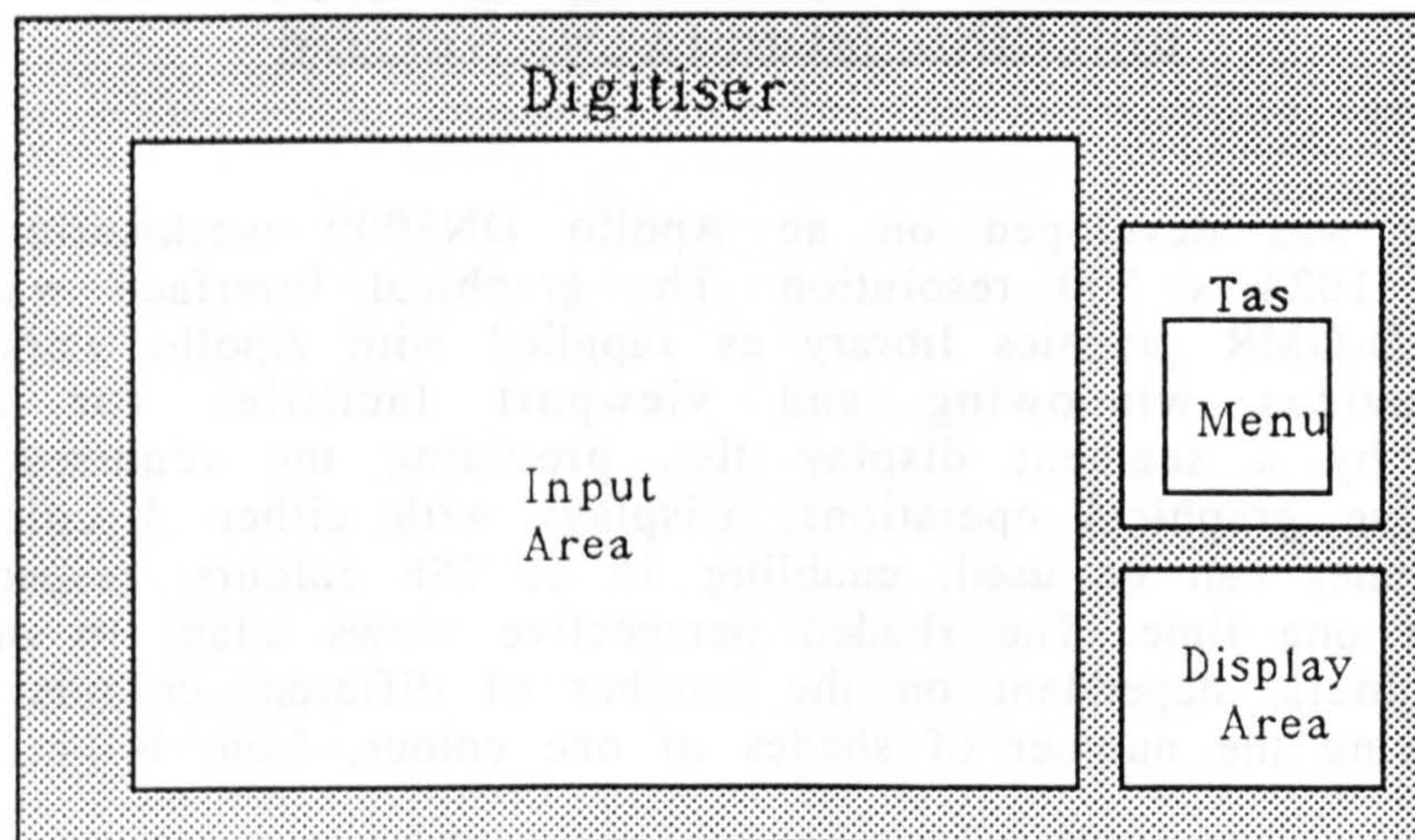
A software driver has been implemented for a range of makes of digitisers c.g. TDS, Calcomp, the dependent interface code for each digitiser type being isolated and therefore easy to implement a different driver for each type. Any size of digitiser can also be used from A3 to A0.

The user interface is controlled by selecting options either from a menu of options placed on the digitiser, or from the screen, using the digitiser mouse. The areas of the digitiser, the positions of which can be varied as required, are shown in figure B-1.

The digitiser menu of options is shown in figure B-2. The options are divided into:

- a) line drawing options, used when positioning the walls and windows of a floor plan.
- b) options, which display a set of sub-options on the screen for further selection. The expansion of these options is shown in figure B-3.





- (i) Digitiser menu: options for creating and editing floor plans are selected from this menu, and also main options to provide access to sub-menus which are displayed on the screen.
- (ii) Display area: area mapped to screen display for selection of options, picking from pictures etc.. It is usually positioned below the digitiser menu.
- (iii) Input area: the area within which the drawing must be fixed for digitising.

**Figure B-1 Digitiser Areas**



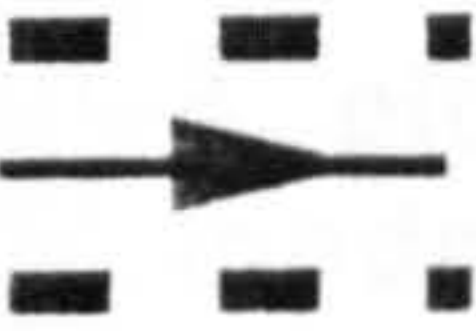



Grid Attach	Grid Parallel	Grid Size	Grid Display	CANCEL
Line Parallel	Line Perpendicular	Line Angle	Line Length	
Datum & Distance	Perpendicular Distance			
			Wall Input Width	Construct'n Line 
Delete Wall / Divide	Divide or Input Wall	Change Wall Type	Display Area	Rescale / Position
Height / Inclination	Building Elements	Windows / Doors	Lower Floor Copy / Display	Input Parameters
Edit Floor	End Edit	Delete Floor		Quit Edit
Areas / Volumes	Zone			Set North
3D Display	Picture Options	Floors Data		EXIT

Figure B-2 Digitiser Menu Options



Edit Floor options

Grid Attach	Grid Parallel	Grid Size	Grid Display	CANCEL
Line Parallel	Line Perpendicular	Line Angle	Line Length	
Datum & Distance	Perpendicular Distance			
			Wall input Width	Construction Line
Delete / Divide	Divide or Input Wall	Change Wall Type	Display Area	Rescale / Position

Heights / Inclinations	Building Elements	Windows / Doors	Lower Floor Copy / Display	Input Parameters
------------------------	-------------------	-----------------	----------------------------	------------------

Heights/ Inclinations
Space select
Wall inclination
Wall height
Point height
Space height
Space Upper Floor
External Space
Define Plane

Building Elements
Name Width Colour
_____
_____
_____
_____
Wall
Floor
Ceiling / Floor Percentage

Window Types
Name Offset H't Width
_____
_____
_____
Place Wall Window
Place Roof Window
Delete Wall Window
Delete Roof Window

Lower Floor Copy/Display
Display all
Copy all
Display Perimeter
Copy Perimeter

Edit Floor	End Edit	Delete Floor		Quit Edit
------------	----------	--------------	--	-----------

Main sub-menus

Areas / Volumes	Zones			Set North
-----------------	-------	--	--	-----------

Areas/Volumes
Space Volume
Wall Area
Floor Area
End Floor

Zones
No
<input type="checkbox"/> 1
<input type="checkbox"/> 2
.....
<input type="checkbox"/> 2
Next
Zones
End
Floor

3D Display	Picture Options	Floors Data		EXIT
------------	-----------------	-------------	--	------

3D Display
Floor Plan
Perspective
Elevation
Plan
Floor Range
Wire Line
Shaded
Hidden Line
Picture Options
Shadow Display

Picture Options
Print Picture
Delete Picture
Zoom
Full Picture
Clear all Pictures
Add Picture
Replace Picture
3D Display

Main Options
No. Floor Description Height Floor Pos'n
_____
_____
_____
_____
Edit Floor
3D Display
Areas/Volumes
Zones
Exit

Shadows
Display Shadows
Sun Position
Day Sequence
Replay Sequence
3D Display

Figure B-3 Digitiser options expanded into sub-options



### EXAMPLES OF GRAPHICAL DISPLAYS

The following figures are examples of the graphical displays of the user interface.

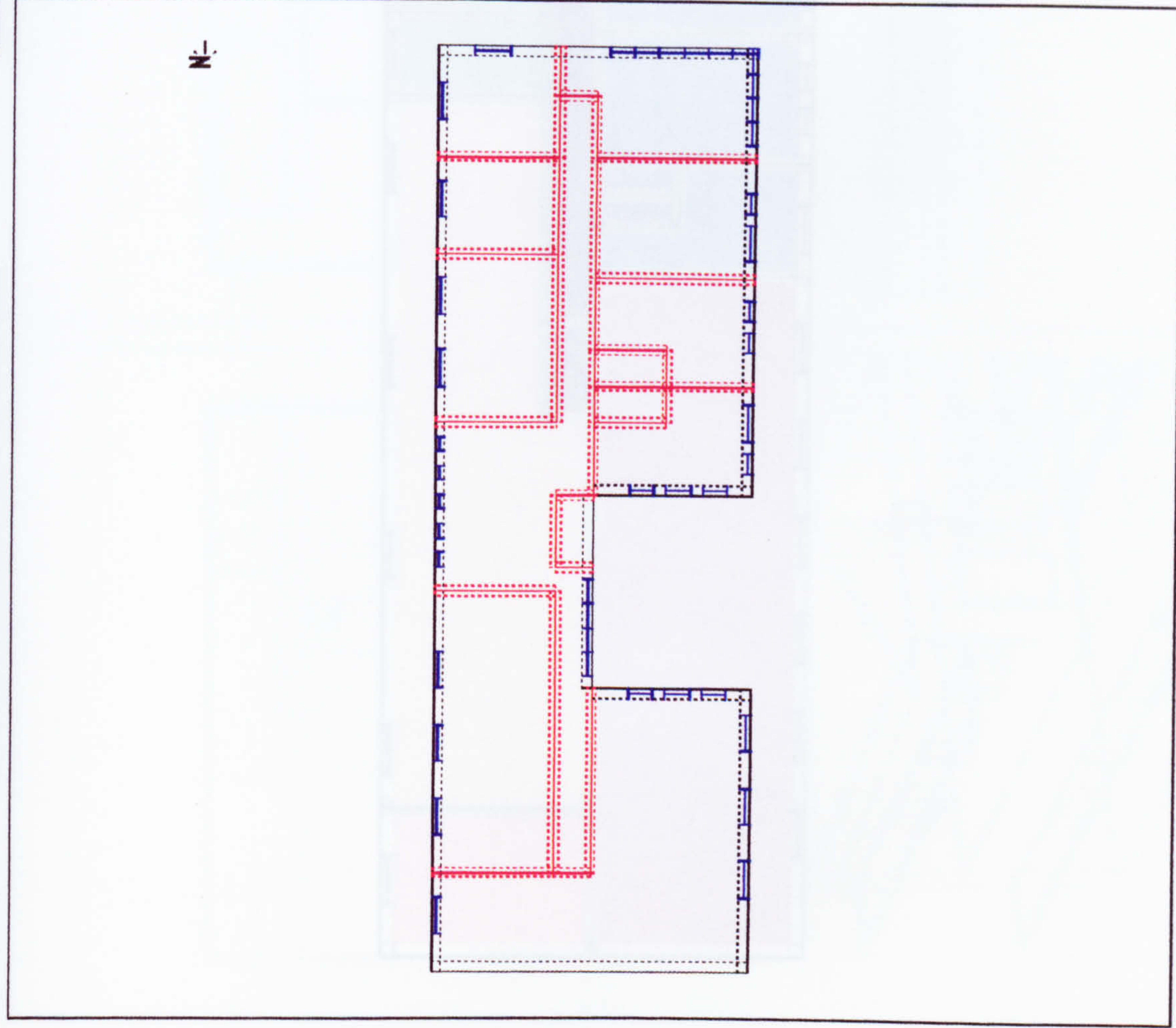
- B-4 Planar graph representation of floor while inputting and modifying walls and windows. The options for positioning windows are shown, with the table of current window types.
- B-5 Assignment of Building Element name to wall constructions, and floor and ceiling faces. The default allocation of 'Internal Wall' is shown by the red highlighted walls.
- B-6 Assignment of zone number to spaces. The current allocation of each space is indicated by the colour key of zones.
- B-7 4 picture display showing views of 3D space representation of building model.
  - Top Left Plan of floor 0 showing variable wall widths according to allocation of Building Element name.
  - Top Right Plan of floor 1.
  - Bottom Left Wireline perspective using zoom function.
  - Bottom Right Hidden Line Elevation.
- B-8 Shaded perspective, with shadows. The sun position used to generate the shadows is shown, in terms of the day of the year and hour, and the calculated azimuth and altitude angles. The position of North and the azimuth of the sun is shown with respect to the building plan.
- B-9 Shaded perspective, also showing shadows cast onto windows.

Figures B-4 to B-9 follow.









Select Wall

[illegible]

Wall

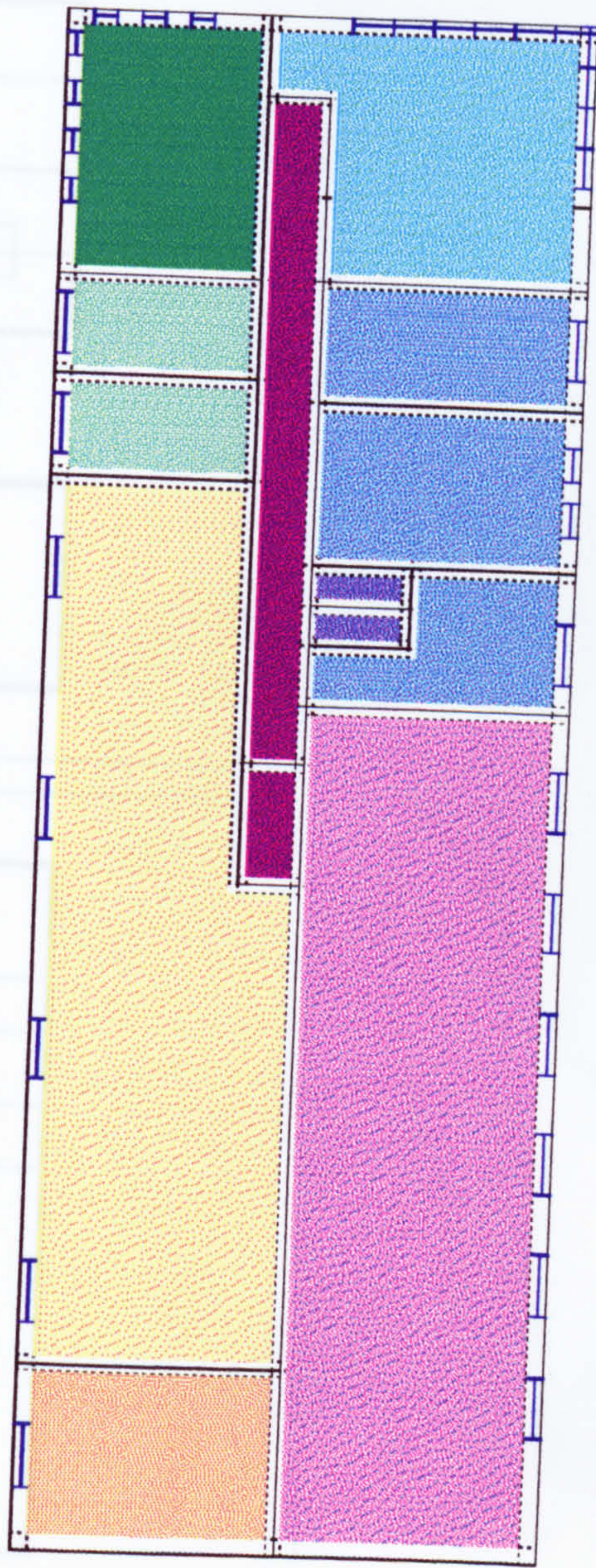
Ceiling/Roof

Floor

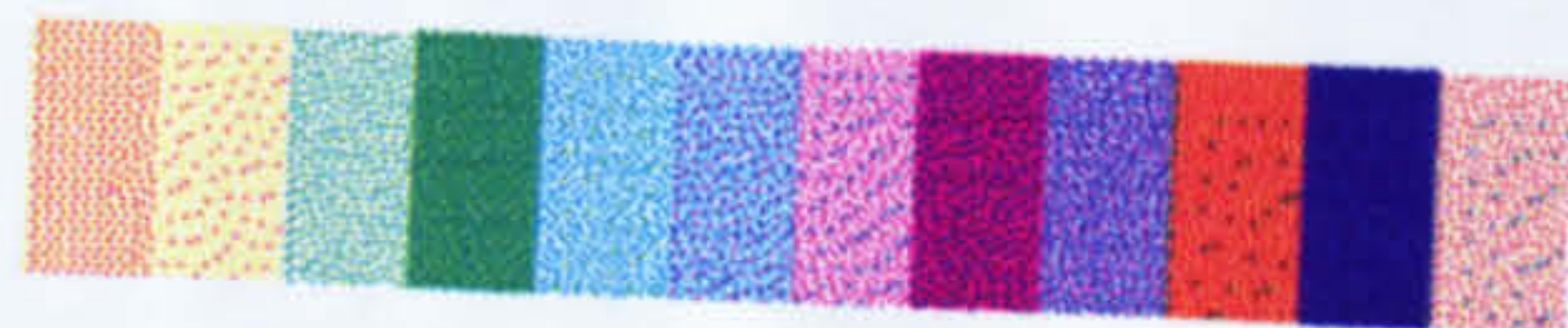
Percentage

## Ground Space





No.	1	2	3	4	5	6	7	8	9	10	11	12
-----	---	---	---	---	---	---	---	---	---	----	----	----

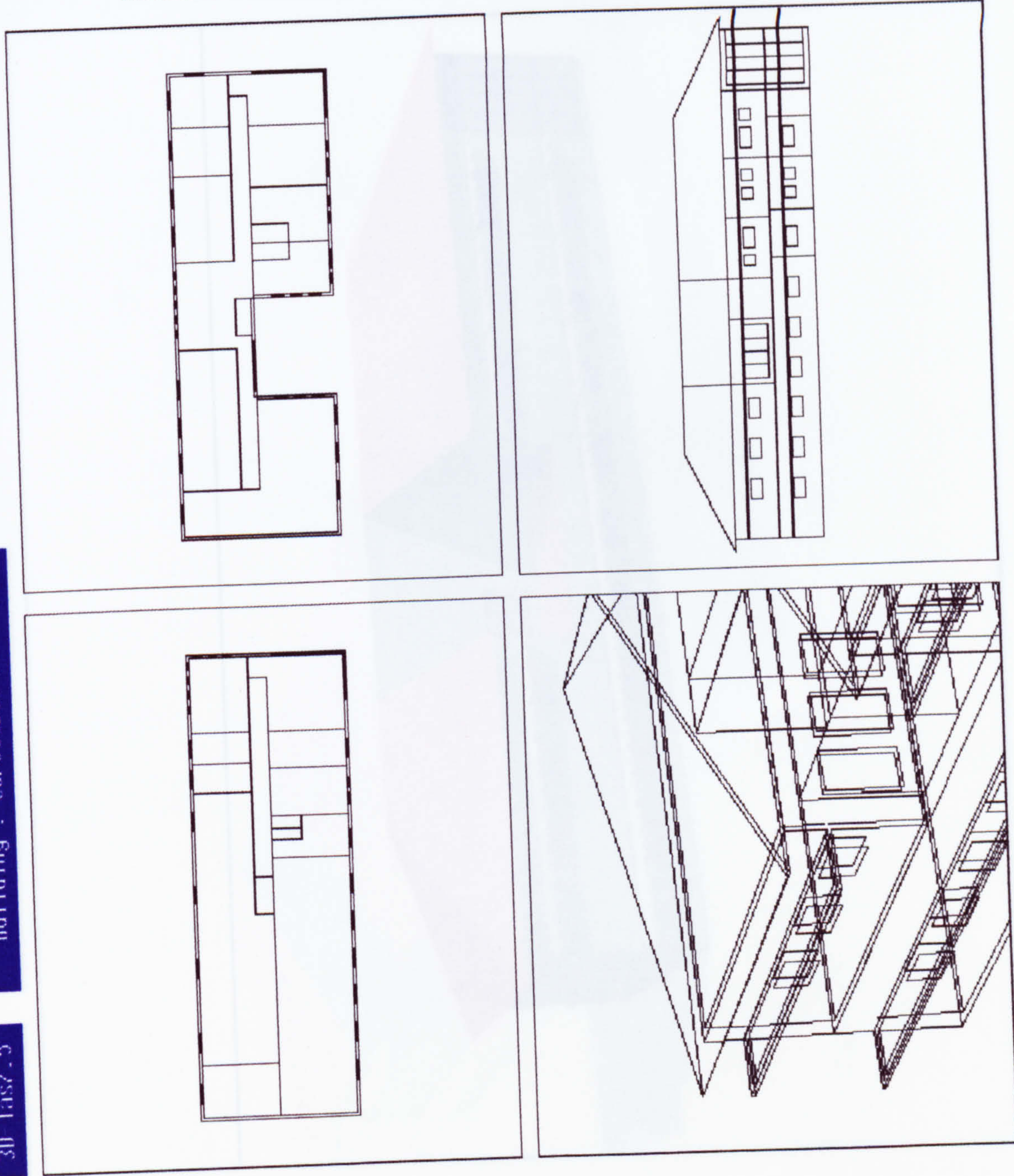


Next Zones

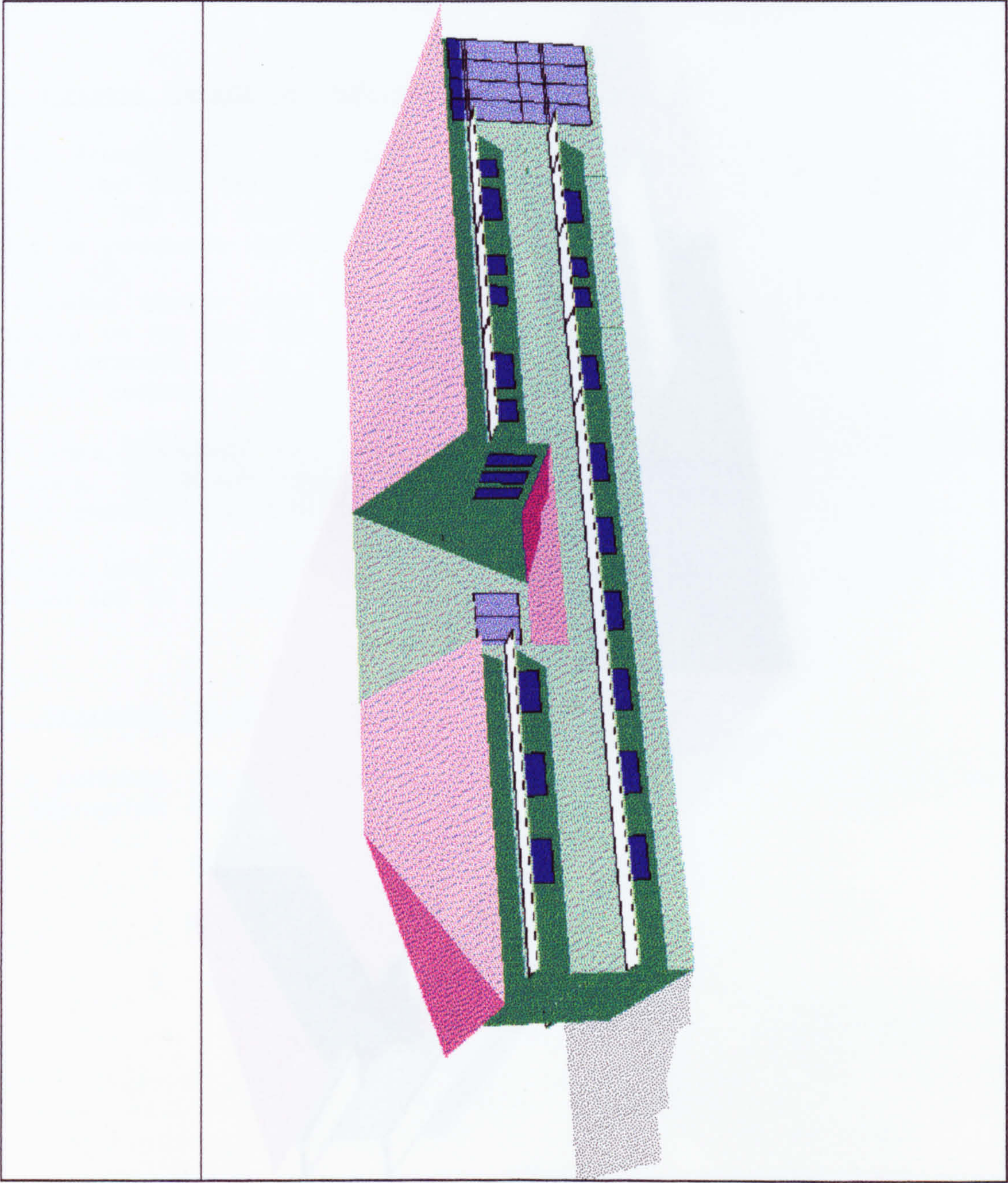
Clear Zones

End Floor









Display Shadows

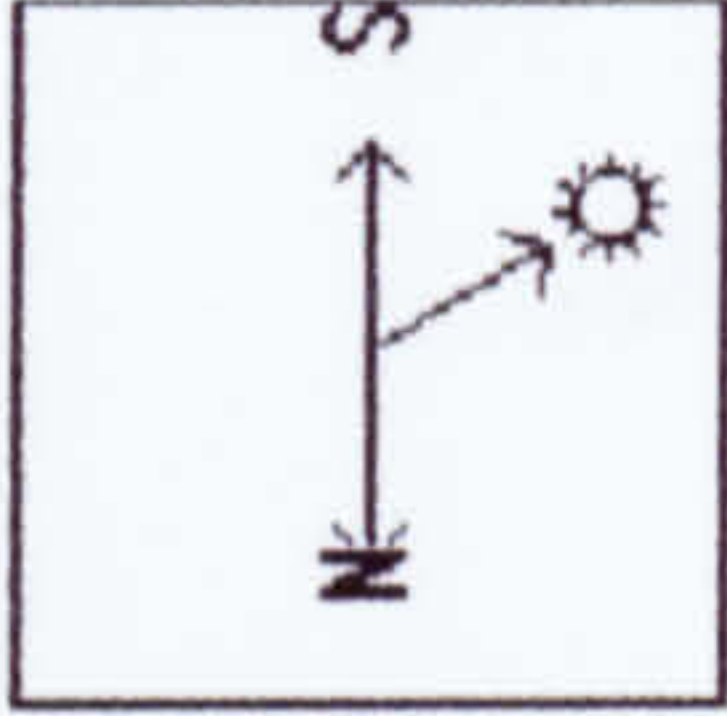
Sun Position

Day	232
Hour	15
Azimuth	239
Altitude	36
Latitude	52.0

Day Sequence

Replay Sequence

3D Display

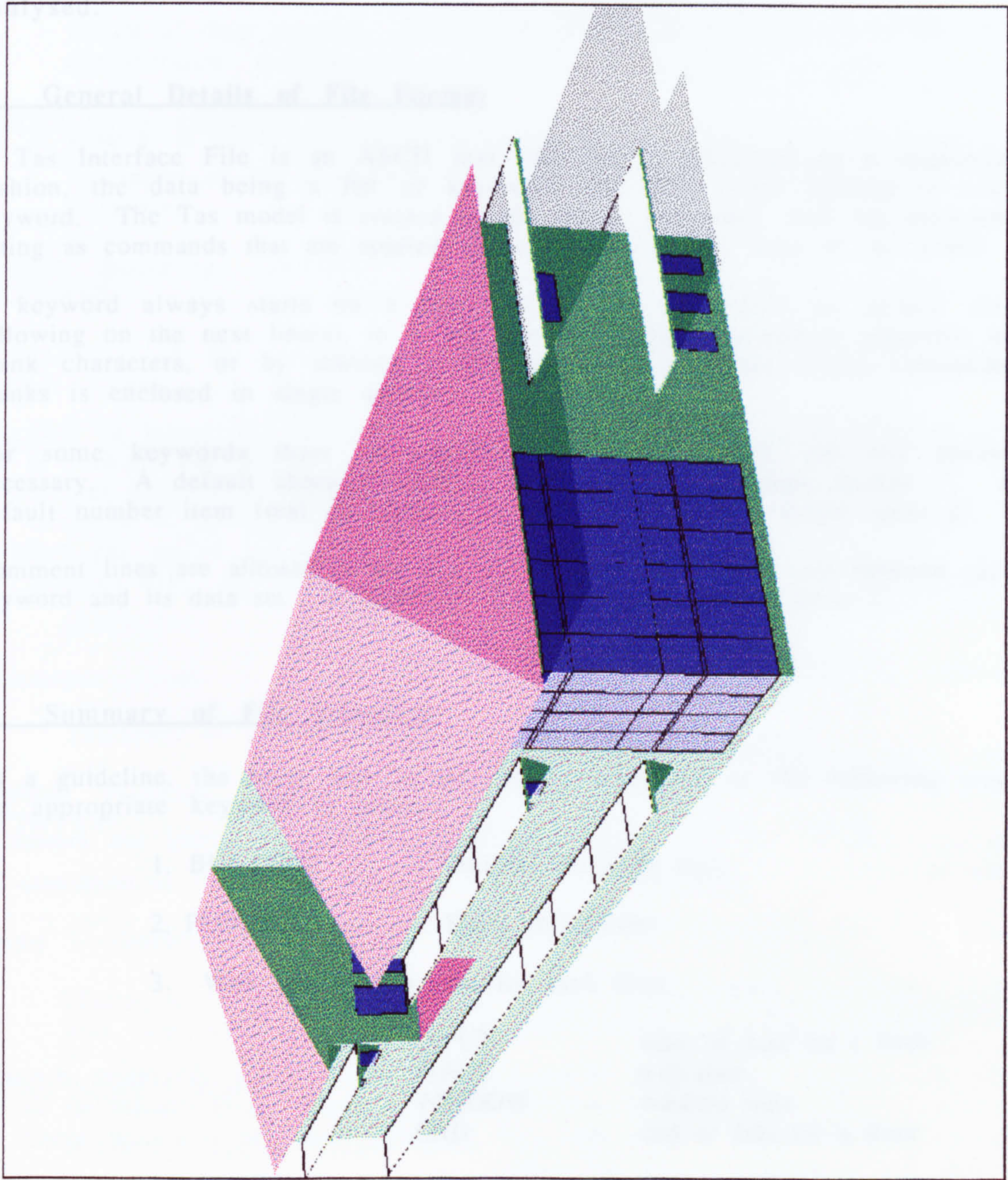




Shadows

Building : eurotube3 rev.03

3D-Ias7.5



Display Shadows

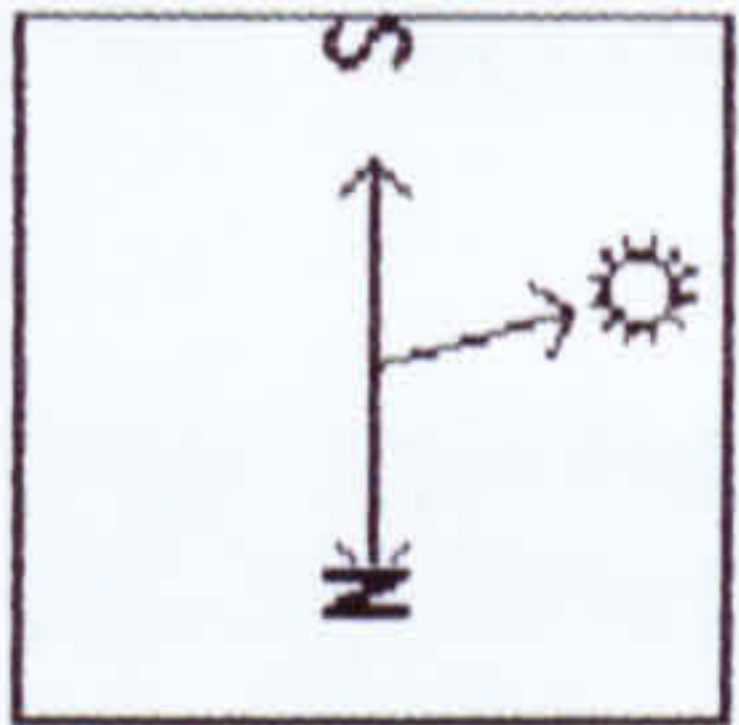
Sun Position

Day	232
Hour	16
Azimuth	253
Altitude	27
Latitude	52.0

Day Sequence

Replay Sequence

3D Display





## APPENDIX C

### TAS INTERFACE FILE FORMAT

This appendix documents the format of a Tas Interface File, which is used to transfer building data from another modelling system to Tas for thermal analysis.

A file of this format contains building data from which a 3D-Tas building model is created. Using Tas, the model can then be displayed and modified. Construction and environmental data is then assigned before the model is analysed.

#### 1. General Details of File Format

A Tas Interface File is an ASCII text file that is processed in a sequential fashion, the data being a list of keywords and data items relating to each keyword. The Tas model is created as the file is processed, with the keywords acting as commands that are applied to act on the current state of the model.

A keyword always starts on a new line in the file, with its related data following on the next line(s), in a free format. Each data item is separated by blank characters, or by starting a new line. A character string containing blanks is enclosed in single quotes.

For some keywords there are default data items which are not always necessary. A default character item is represented by 2 single quotes: ". A default number item (real or integer) is represented by the single letter D.

Comment lines are allowed in the file at the start of the file and between each keyword and its data set. A comment line must start with the letter C.

#### 2. Summary of File Structure

As a guideline, the basic data in the file is structured in the following way, the appropriate keyword is given:

1. BUILDING           - General Building data.
2. FLOOR             - Floor definitions
3. Wall and window data for each floor
 

INPUT	-	start of data for a floor
WALL	-	wall data
WINDOW	-	window data
END	-	end of data for a floor
4. SAVE              - Save data to 3D-Tas model file.
5. EXIT              - end data in file.

### 3. Description of Keywords

Each keyword is now described with its related ordered data items. The type of the data item is given as character, real or integer and any items that can be defaulted are indicated.

Rules are also given to define the order in which the keyword must appear in the file in relation to other keywords, where necessary.

#### Identification of Building

**BUILDING**                      building    name,    description

This should be the first keyword in the file.

building name - character

Maximum of 10 characters, no blank characters, no full stops. It is used as the TAS model name.

Description                - character

Maximum of 40 characters

#### Tas Model Revision number

**REVISION**                      number

The model revision number is used when creating the Tas model as indicated by the 'SAVE' keyword. The revision number is set to 1 if this keyword is not given. This keyword may also be used to indicate that an existing revision of Tas model of the building is to be modified, add an extra floor for example. In this case, this keyword must appear after 'BUILDING' but before any building data is input (before FLOOR, ELEMENT, WINTYP or INPUT). The existing Tas model is then retrieved from file.

number    -    integer

#### Set units to metres or feet

**UNITS**                              units    indicator

Defines units of all real values in following keywords. Units are metres if this keyword is not present.

units indicator - character 'M' for metres or 'F' for feet.

#### Wall Linking Distance

**LINKDIST**                      distance

For the purpose of forming enclosed spaces, this is the maximum distance between 2 walls that are to be considered to be joined. It is usually set to the maximum wall width. This value will also determine the minimum length of a wall that is accepted in the Tas model.

distance - real

**Define Floor**  
**FLOOR**

floor number, z datum, description

FLOOR must be present before the INPUT of the given floor or any lower floors which have walls whose height will be greater than the z datum.

floor number - integer

Number in range -10 to 50.

z datum - real

Datum of floor position. This is the position of the top side of the floor construction element.

description - character

Maximum of 20 characters. A description is not necessary and can be defaulted

**Define Building Element**  
**ELEMENT**

name, thickness

A maximum of 30 elements are allowed in a Tas model, including the Tas default names.

name - character

Maximum of 15 characters. This name is referred to in the INPUT and WALL data, and therefore this data must appear first. Element names for ceiling and floors should differ in the first 7 characters. This name can be one of the Tas default element names to assign a thickness.

thickness - real

Thickness of the wall, ceiling or floor element.

**Define Window Type**  
**WINTYP**

name, offset height, height, width

A maximum of 20 window types are allowed in a Tas model.

name - character

Maximum of 20 characters. This name is referred to in the WINDOW data, and therefore this data must appear before the WINDOW data.

offset height - real

Height of window bottom above floor datum



height - real

Height of window above window bottom

width - real

Width of window. If this item is defaulted the width of the window is determined by 2 co-ordinates given in the WINDOW detail.

### Start of Floor Data

**INPUT**                      Floor number, floor element name,  
                                 ceiling element name

Before this keyword is accepted it is checked that the floor data so far define a set of floors with sequentially increasing floor numbers and z datums.

floor number - integer

A floor number that has previously been defined by FLOOR data, and is the next in the sequence to be input.

floor element name - character

Name, previously defined in ELEMENT data, that will be assigned to all floor constructions of spaces on this floor of the building. If this name is not given, a default building element will be assigned, either 'Ground Floor' or 'Upper Floor'.

ceiling element name - character

Name previously defined in ELEMENT data, that will be assigned to all ceiling constructions of spaces on this floor of the building. If this name is not given, the default building element 'Ceiling' will be assigned.

### Position a Wall

**WALL**                      line type, element name  
                                 x1,y1, left height1, right height1,  
                                 x2.y2, left height2, right height2

This data must appear between an INPUT-END pair. A wall is not accepted if it is shorter than the linking distance.

line type - integer

This indicates which line of the wall the end-points refer to, and where the width of the wall, given in the ELEMENT definition, is positioned.

element name - character

Name, previously defined in ELEMENT data, to be assigned to the wall. If this name is not given, a default building element name will be assigned, either 'External Wall' or 'Internal Wall' depending on whether the wall lies on the external perimeter or the floor or not.

x1,y1 - 2 reals

Co-ordinates of first point of wall.

left height1 - real

Height at 1st end of wall, right side, above current floor datum

x2,y2 - 2 reals

Co-ordinates of 2nd end-point of wall.

left height2 - real

Height at 2nd end of wall, left side, above current floor datum.

right height2 - real

Height at 2nd end of wall, right side, above current floor datum.

Heights can be defaulted, when the height will be the difference between the z datums of the current and the next floor.

#### Position a Wall Window

WINDOW                      type name, x1,y1    x2,y2

A window must be positioned within a wall. A 'Window' can be used to define any rectangular area placed within a wall, and not necessarily transparent for example a door.

type name - character

Name, previously defined in WINTYP data

x1,y2 - 2 reals

Co-ordinates of 1st end-point of window

x2,y2 - 2 reals

If the width in the window type is defaulted : co-ordinates of 2nd end-point of window. If the width is given in the window type : direction vector of window line from 1st end-point.

### End of Floor END

This indicates the end of wall and window data for a floor. The 3D floor is created from the 2D wall and window data to form enclosed spaces.

Various checks are made on the wall data before the 3D model of the floor can be created:

- a) There should be no disconnected walls in an internal space or external to the building perimeter. This means that both ends of each wall must be connected to another wall. Any such walls will be deleted from the 3D model.
- b) If a default height is set for any walls and there is no 'FLOOR' data for the next floor, a 3D model can not be created. In this case the floor input fails and a 3D model is not created and saved to file.
- c) It is checked that a planar ceiling is defined for each space, using the assigned wall heights. If this is not the case the default floor to floor height is assigned to the walls surrounding the space.

### Save to File SAVE

A Tas model is created with the given building name and current revision number. This keyword can appear at any time in the file, as many times as necessary if different revisions of the building model are required at different stages in the creation.

If SAVE is present before END of a floor, the 3D model of the floor is not created, but on retrieving the model within Tas, all of the disconnected walls are still present. This could be useful if additional enclosed spaces need to be defined for the analysis by using these disconnected walls and entering extra walls within Tas.

### End of File EXIT

This indicates the end of the data for the building.

Examples of Tas Interface Files

Example 1            Building with 1 floor, 2 rooms,  
                         using default building elements

```
BUILDING
EXAMPLE1
'Basic Building - 1 Floor, 2 rooms'
FLOOR
1            1.0            'First Floor'
INPUT
1 " "
WALL
2 "
0.0           0.0           3.0           3.0
0.0           10.0          3.0           3.0
WALL
2 "
0.0           10.0          3.0           3.0
10.0          10.0          3.0           3.0
WALL
2 "
10.0          10.0          3.0           3.0
10.0          0.0           3.0           3.0
WALL
2 "
10.0          0.0           3.0           3.0
0.0           0.0           3.0           3.0
WALL
1 "
5.0           0.0           3.0           3.0
5.0           10.0          3.0           3.0
END
SAVE
EXIT
```

Example 2      Building with 1 floor, 2 rooms, 2 windows and a door,  
                  using default building elements

BUILDING  
EXAMPLE2

'1 Floor, 2 rooms and windows'

FLOOR

1                    1.0                    'First Floor'

INPUT

1 " "

WALL

2 "

0.0                    0.0                    3.0                    3.0

0.0                    10.0                    3.0                    3.0

WALL

2 "

0.0                    10.0                    3.0                    3.0

10.0                    10.0                    3.0                    3.0

WALL

2 "

10.0                    10.0                    3.0                    3.0

10.0                    0.0                    3.0                    3.0

WALL

2 "

10.0                    0.0                    3.0                    3.0

0.0                    0.0                    3.0                    3.0

WALL

1 "

5.0                    0.0                    3.0                    3.0

5.0                    10.0                    3.0                    3.0

WINTYP

                  WINDOW1                    0.5                    2.0                    D

WINTYP

                  DOOR                    0.0                    3.0                    D

WINDOW

                  WINDOW1  
0.0                    1.0                    0.0                    4.0

WINDOW

                  WINDOW1  
1.0                    0.0                    4.0                    0.0

WINDOW

                  DOOR  
0.0                    7.0                    0.0                    8.0

END

SAVE

EXIT



Example 3 Building with 2 floors, assigned building elements  
and varying heights of walls on 2nd floor

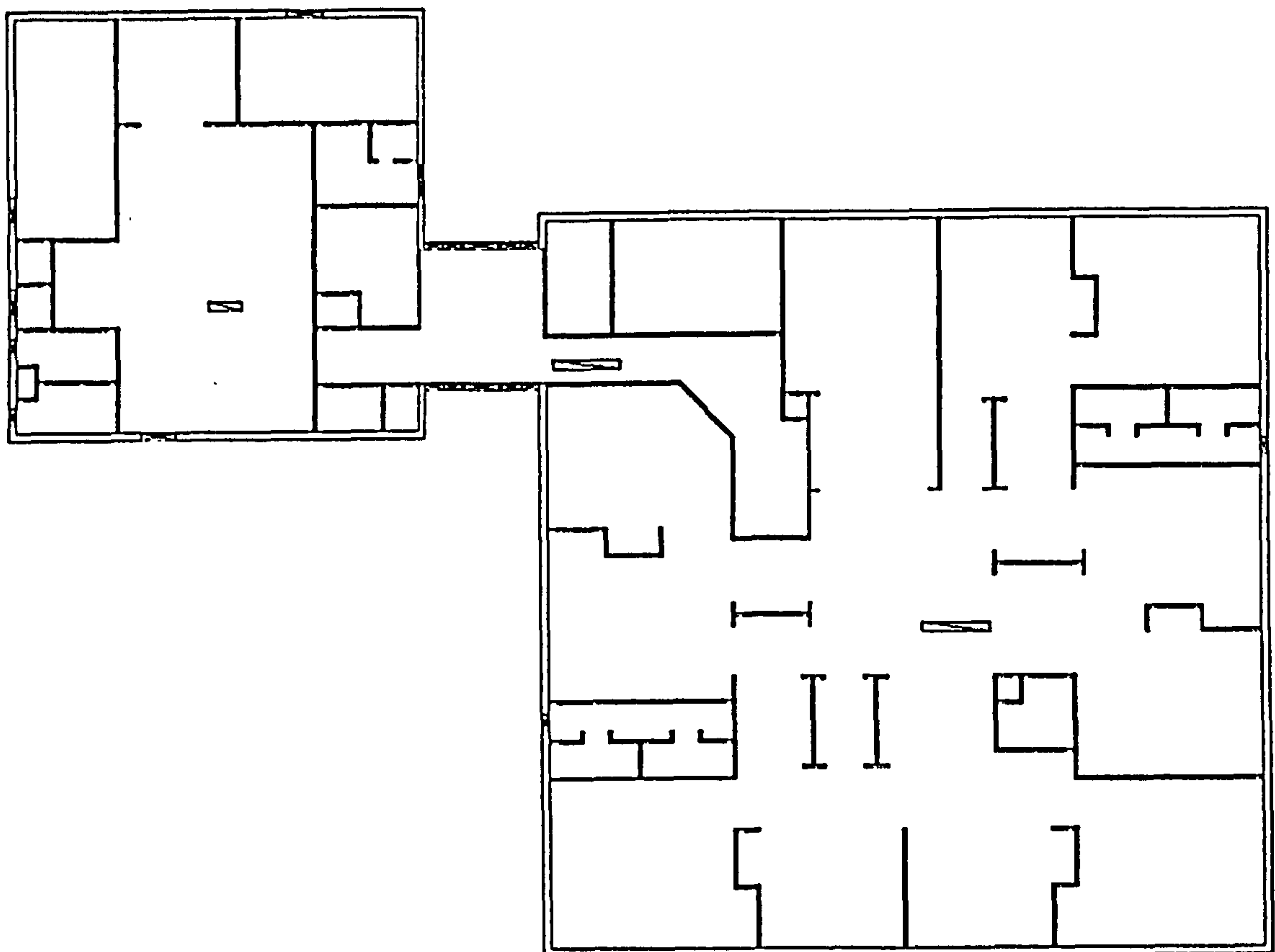
BUILDING  
EXAMPLE3  
'2 floors, Building Elements assigned'  
FLOOR  
FLOOR 1 1.0 'Ground Floor'  
FLOOR 2 4.0 'First Floor'  
C  
C Assign width to default building elements  
ELEMENT 'External Wall' 0.4  
ELEMENT Ceiling 0.2  
C  
C Define new building elements  
ELEMENT EXWALL 0.2  
ELEMENT INWALL 0.1  
ELEMENT ROOF 0.1  
ELEMENT CONCRETE 0.0  
C  
C Start of input of floor 1 use of default ceiling element  
C  
C  
INPUT  
WALL 1 CONCRETE "  
WALL 2 EXWALL  
0.0 0.0 D D  
0.0 10.0 D D  
WALL 2 EXWALL  
0.0 10.0 D D  
10.0 10.0 D D  
WALL 2 EXWALL  
10.0 10.0 D D  
10.0 0 D D  
WALL 2 EXWALL  
10.0 0.0 D D  
0.0 0.0 D D  
WALL 1 INWALL  
5.0 0.0 D D  
5.0 10.0 D D  
END

C  
C     Start of input of floor 2 -use default floor and wall elements  
C     2 spaces defined-one with horizontal roof at height of 3 mtrs  
C     and one with roof sloping from 2 metres down to 1 metre.  
C

INPUT	2	"	ROOF
WALL	2	"	
0.0	0.0	3.0	3.0
0.0	5.0	3.0	3.0
WALL	2	"	
0.0	5.0	2.0	2.0
0.0	10.0	1.0	1.0
WALL	2	"	
0.0	10.0	1.0	1.0
10.0	10.0	1.0	1.0
WALL	2	"	
10.0	10.0	1.0	1.0
10.0	5.0	2.0	2.0
WALL	2	"	
10.0	5.0	3.0	3.0
10.0	0.0	3.0	3.0
WALL	2	"	
10.0	0.0	3.0	3.0
0.0	0.0	3.0	3.0
WALL	2	"	
0.0	5.0	2.0	3.0
10.0	5.0	2.0	3.0
END			
SAVE			
EXIT			

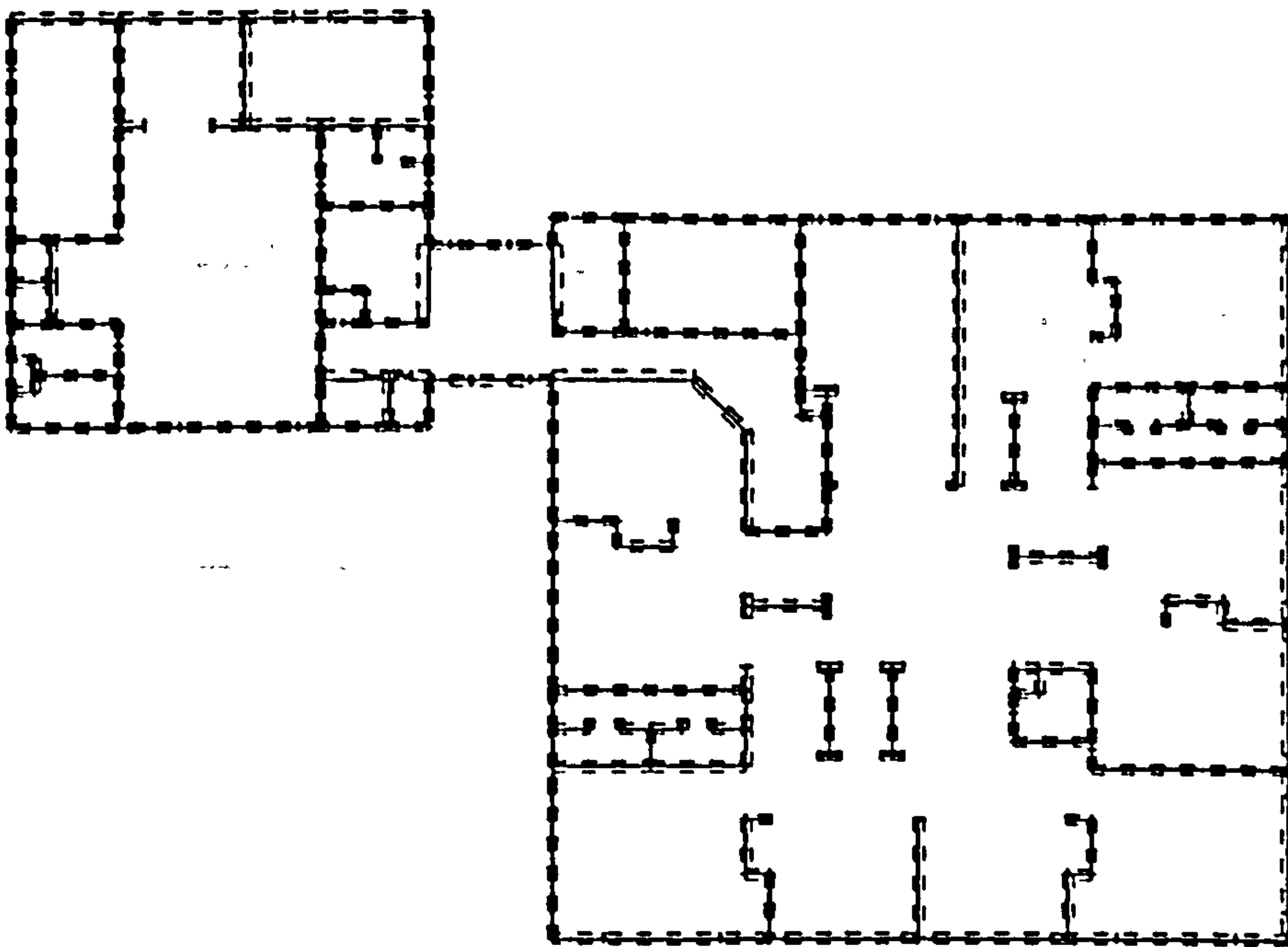
### Example of a Transferred Building Model

The following figures show the representation of a building within the GABLE CAD system, and the resulting model created within Tas.

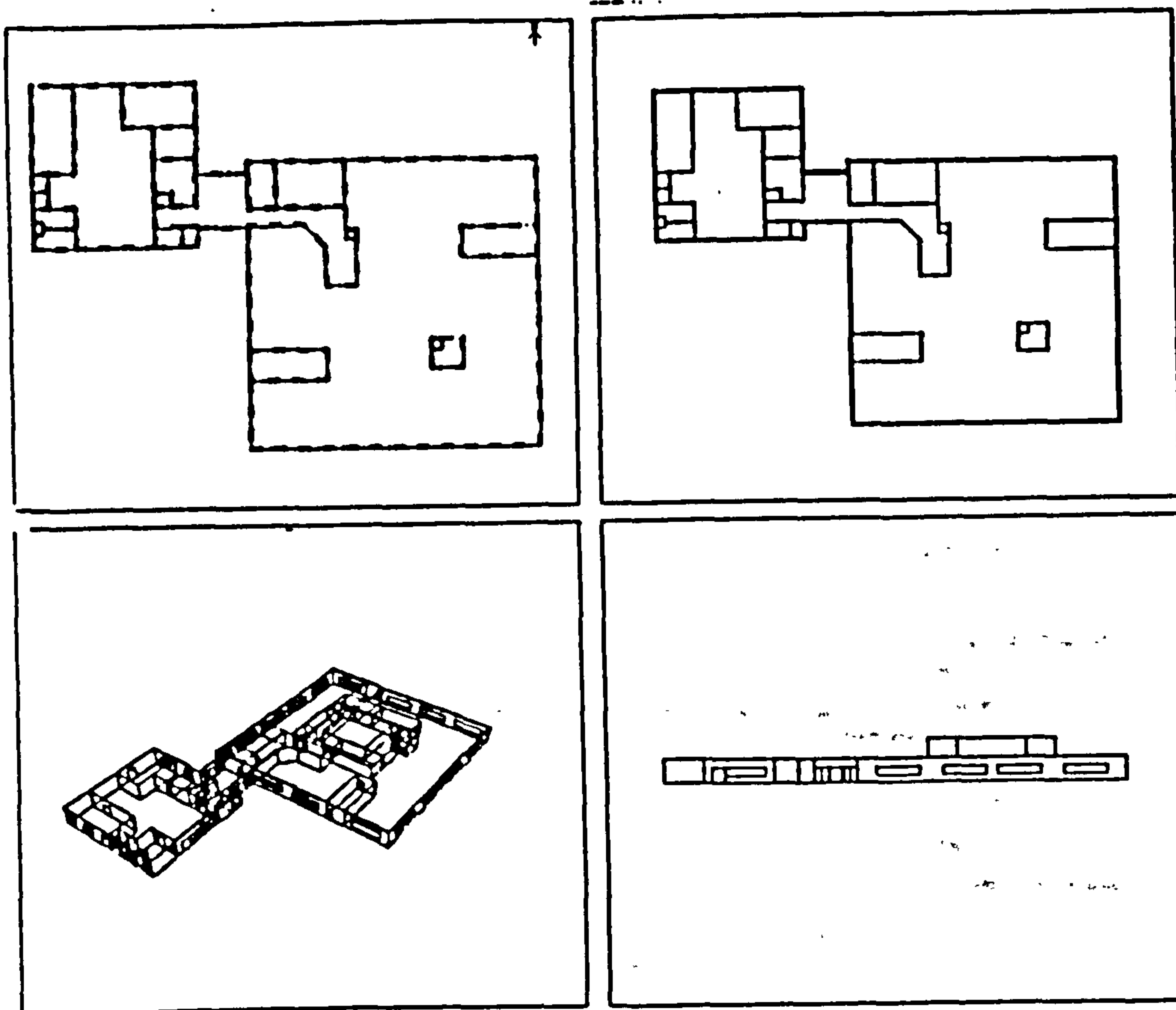


**Figure C-1**      **GABLE display of floor 1 of building**



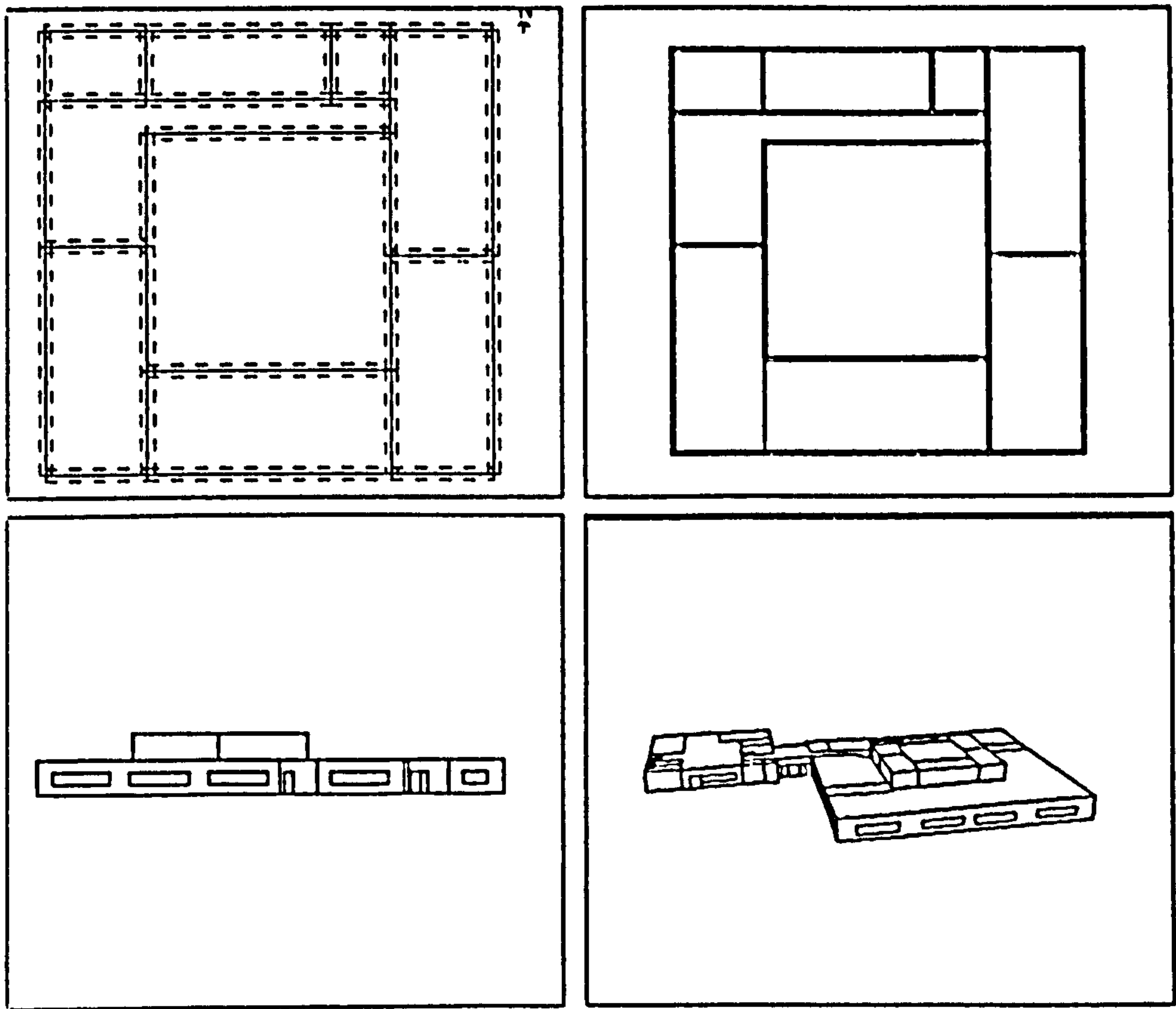


**Figure C-2**      **Floor 1 created from transferred data within Tas**



**Figure C-3**

<b>Top Left:</b>	<b>Floor 1 after wire edge removed, and used to create space representation</b>
<b>Top Right:</b>	<b>Plan of space representation of floor 1</b>
<b>Bottom Left:</b>	<b>Wireline perspective of space representation</b>
<b>Bottom Right:</b>	<b>Hidden line elevation</b>



**Figure C-4**    **Top Left:**      **Floor 2 created from transferred data**  
**Top Right:**    **Plan of space representation of**  
                                 **floor 2**  
**Bottom Left:**   **Hidden line elevation**  
**Bottom Right:** **Hidden line perspective of floors 1**  
                                 **and 2**